一线SAP专家扛鼎之作
为SAP ABAP开发提供解决方案

孙东文 郭娟 等编著

SAP

Web Dynpro For ABAP开发技术详解–高端应用

SAP Web Dynpro For ABAP Detailed Development Technology-High-end Application







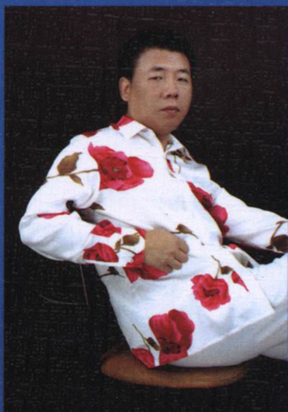
机械工业出版社
CHINA MACHINE PRESS

VISIT…

LANZAROTE

Caliente.COM

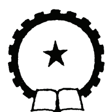
孙东文



孙东文，笔名：东方先生，SAP顾问，山东沾化人。毕业于大连舰艇学院并获得学士学位，从事SAP推广、实施、维护及教学近10年。

SAP Web Dynpro For ABAP 开发技术详解——高端应用

孙东文　郭　娟　郭　欢　张　岩　编著



机械工业出版社

本书主要介绍了与SAP Web开发技术——Web Dynpro For ABAP 相关的集成应用和高级应用，还讲解了 Web Dynpro For ABAP 高端应用的各个技术难点中各元素之间的关联及制约关系。

本书结合实例讲解了在 Web Dynpro 组件中如何集成 Flex，如何存储和访问文件，Web Dynpro 如何在 EP（Enterprise Portal）中展现及数据交互，如何继承 Adobe Form 这种应用极广的电子表格以及与之相关的数据的存取，广泛应用于 SAP 系统的 ALV 表格如何在 Web Dynpro For ABAP 中应用集成以及更多异于一般 ALV 列表的功能。此外，本书还介绍了在 Web Dynpro 程序中如何创建和使用辅助类，如何创建和使用服务调用，与用户交互时如何使用对话框、输入帮助、文本帮助及消息，在考虑比较复杂的 Web Dynpro 应用之间交互时如何使用 Suspend 和 Resume 调用等。

本书深入剖析了 Web Dynpro For ABAP 的关键技术，可作为 Web Dynpro For Java 开发人员深入学习 SAP 系统的参考书。

图书在版编目（CIP）数据

SAP Web Dynpro For ABAP 开发技术详解：高端应用 / 孙东文等编著. —北京：机械工业出版社，2016.8
ISBN 978-7-111-54749-5

Ⅰ. ①S… Ⅱ. ①孙… Ⅲ. ①企业管理－应用软件－软件开发
Ⅳ. ①F272.7-39

中国版本图书馆 CIP 数据核字（2016）第 210190 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策划编辑：丁 诚 责任编辑：丁 诚 范成欣
责任校对：张艳霞
责任印制：李 洋

三河市宏达印刷有限公司印刷

2016 年 9 月第 1 版 • 第 1 次印刷
184mm×260mm • 17 印张 • 415 千字
0001－3000 册
标准书号：ISBN 978-7-111-54749-5
定价：59.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
服务咨询热线：（010）88361066
读者购书热线：（010）68326294
（010）88379203

网络服务
机 工 官 网：www.cmpbook.com
机 工 官 博：weibo.com/cmp1952
教育服务网：www.cmpedu.com
金 书 网：www.golden-book.com

封面无防伪标均为盗版

前　　言

随着 SAP 在中国企业中的广泛应用，SAP 实施及运维行业也获得了高速增长，SAP 相关技术也得到了广泛的应用，这其中不乏 SAP Web 开发的高端技术。

作者参考 SAP 官方资料和加上作者多年积累的项目经验对 ABAP 开发技术做了整理和提炼，围绕 Web Dynpro For ABAP 这样一门 SAP Web 编程的高级开发技术编写了本书。

以往的书籍仅对 ABAP Web 编程做一下概念上的剖析，能够提及原理、架构及实现的图书少之又少。本书的特点在于对各技术原理、实施前提、实施步骤、实施结果作了详尽的说明，更有关于 SAP WDA 开发类和服务类的相关介绍，以帮助那些渴望提升技术水平的 SAP 开发人员。

学习 SAP Web 开发技术，开发者不仅需要提高自身的水平（如从面向过程编程到面向对象编程的转变，从面向对象的语言开发到基于 MVC 理论的 Web 开发之间的转变及对 MVC 理论下的数据关系映射的了解），还需要熟悉 SAP 相关集成应用，（如 Web Dynpro 组件中如何集成 Flex、系统文件、EP（Enterprise Portal）、Adobe Form 电子表格等），更需要 SAP 开发者对各种高级概念的掌握（如如何创建和使用辅助类，如何创建和使用服务调用，与用户交互时如何使用对话框等）。

参与本书编写的有孙东文（笔名：东方先生）、郭娟、郭欢、张岩。由于编者水平有限，书中难免有疏漏和不足之处，请读者批评指正。

编　者

目　　录

前言
第 1 章　**Web Dynpro For ABAP 的由来** …… 1
1.1　ABAP …… 1
1.2　NetWeaver …… 1
1.3　Web Dynpro …… 2
1.3.1　Web Dynpro 的双重目的 …… 2
1.3.2　Web Dynpro 技术要点 …… 3
第 2 章　**集成 Adobe Flash Islands** …… 4
2.1　Flash Island UI 元素的数据传输 …… 4
2.2　Flash Island UI 元素的事件 …… 5
2.3　编辑 Adobe Flex 应用程序 …… 5
2.4　编辑 Web Dynpro 应用程序 …… 9
第 3 章　**文件操作** …… 18
第 4 章　**门户集成（Portal Integration）** …… 25
4.1　绑定到门户的准备 …… 25
4.2　门户中 Web Application 集成 …… 25
4.3　门户事件（Portal Events） …… 29
4.3.1　触发门户事件 …… 29
4.3.2　注册并处理事件 …… 31
4.4　事件应用实例 …… 32
4.4.1　创建应用程序 …… 32
4.4.2　创建 iView 及页面 …… 40
4.5　门户导航（Portal Navigation） …… 49
4.5.1　基于对象的导航（OBN） …… 49
4.5.2　绝对地址导航 …… 51
4.5.3　相对地址导航 …… 53
4.5.4　Resume Plugs 和门户导航 …… 54
4.6　导航应用实例 …… 54
4.6.1　业务对象创建 …… 54
4.6.2　应用程序创建 …… 58
4.6.3　应用程序导航测试 …… 65
4.7　工作保护模式（Work Protect Mode） …… 67
第 5 章　**集成表单（Integrating Forms）** …… 69
5.1　集成表单条件 …… 70
5.2　在 Web Dynpro 应用中集成 PDF 表单 …… 71

5.3 插入互动表单 ······ 72
5.3.1 在 Web Dynpro 应用中使用 Adobe Forms ······ 72
5.3.2 在 Web Dynpro 应用中新建 Adobe Forms ······ 76
5.4 插入已有 PDF 表单（MIME 对象） ······ 84
5.5 表单集成中的数字签名（Digital Signatures） ······ 86
5.6 支持 Adobe 库中的 UI 元素 ······ 86
5.7 交互式表单应用 ······ 88
5.8 基于 ABAP 数据字典接口表单 ······ 88
5.9 隐藏 Adobe Toolbar ······ 89
第 6 章 ALV 在 Web Dynpro ABAP 中的应用 ······ 90
6.1 在应用程序中集成 ALV ······ 90
6.1.1 组件使用定义 ······ 92
6.1.2 供应数据 ······ 92
6.1.3 使用 ALV 视图 ······ 94
6.1.4 ALV 配置模型对象 ······ 94
6.1.5 ALV 配置模型 ······ 95
6.1.6 类图（Class Diagram） ······ 96
6.2 管理 ALV 输出领域 ······ 96
6.2.1 ALV 标题（ALV Output Header） ······ 96
6.2.2 字段 ······ 98
6.2.3 列 ······ 99
6.2.4 页脚和滚动条 ······ 103
6.2.5 页眉与页脚区域 ······ 105
6.2.6 设定对话框 ······ 110
6.3 ALV 输出的外观 ······ 111
6.3.1 将属性分配给列和单元格 ······ 111
6.3.2 ALV 输出、列和单元格的大小 ······ 112
6.3.3 个别区域的可见性 ······ 114
6.3.4 ALV 输出、列和单元格的颜色 ······ 114
6.3.5 文本属性 ······ 115
6.3.6 列和行之间的网格 ······ 116
6.3.7 树状层次结构表 ······ 116
6.3.8 表数据作为业务图形输出 ······ 118
6.3.9 空输出的替换文本 ······ 121
6.4 预定义标准 ALV 功能 ······ 121
6.4.1 排序 ······ 121
6.4.2 过滤器 ······ 124
6.4.3 计算（合计） ······ 126
6.4.4 配置和个性化 ALV 输出 ······ 129

6.4.5 输出……130
6.5 功能、交互和事件……134
6.5.1 提供 ALV 功能……134
6.5.2 没有数据变化的交互与处理……141
6.5.3 可编辑 ALV 的互动处理……143
6.5.4 为用户提供帮助……147
6.6 接口控制器的方法和事件……149
6.6.1 DATA_CHECK……149
6.6.2 GET_CONFIG_DATA……149
6.6.3 GET_MODEL……151
6.6.4 GET_MODEL_EXTENDED……151
6.6.5 SET_FOCUS……152
6.6.6 SET_DATA……152
6.6.7 ON_AFTER_CONFIG……152
6.6.8 ON_CELL_ACTION……152
6.6.9 ON_CLICK……153
6.6.10 ON_DATA_CHECK……153
6.6.11 ON_FUNCTION……153
6.6.12 ON_LEAD_SELECT……153
6.6.13 ON_SELECT……153
6.6.14 ON_STD_FUNCTION_AFTE……154
6.6.15 ON_STD_FUNCTION_BEFO……154
6.7 在 Web Dynpro 应用中使用 ALV 的简单实例……154
第 7 章 高级概念……164
7.1 组件辅助类……164
7.2 Web Dynpro 应用中的服务调用（Service Call）……170
7.2.1 创建一个 Service Call……171
7.2.2 使用 Service Call……175
7.3 使用对话框……180
7.3.1 同一组件中调用对话框……181
7.3.2 调用使用组件的对话框……183
7.3.3 调用确认对话框……184
7.4 输入帮助……189
7.4.1 OVS 输入帮助……193
7.4.2 自定义输入帮助……195
7.4.3 选择标准……196
7.4.4 编辑 Web Dynpro 应用程序……197
7.5 ABAP Web Dynpro 应用程序的帮助文本……214
7.5.1 Explanation UI 元素……216

7.5.2 Explanation 属性 ······ 218
7.5.3 典型 F1 帮助 ······ 220
7.6 消息 Messages ······ 221
7.6.1 消息弹出 ······ 222
7.6.2 消息日志中的消息集成 ······ 223
7.6.3 配置消息 ······ 224
7.6.4 重用组建和 MessageArea 元素 ······ 225
7.6.5 编辑 Web Dynpro 应用程序 ······ 225
7.7 Suspend 和 Resume 调用 ······ 234
7.8 处理 Web 图标 ······ 237
附录 A Web Dynpro For ABAP 系统类 ······ 239
附录 B 服务类（Service Classes） ······ 259

第 1 章　Web Dynpro For ABAP 的由来

Web Dynpro 是传统 Dynpro 编程方法在互联网时代的扩展。SAP 为 Web Dynpro 提供了两种开发语言（ABAP 和 Java），于是就有了 Web Dynpro for ABAP 和 Web Dynpro for Java。

Web Dynpro 与 SAP 以往的设计范例完全不同，代表了开发基于 Web 的 ERP Application 取得的重大突破。基于 MVC 设计模式的 SAP Web Dynpro for ABAP 在开发基于表单的用户界面中具有革命性的意义。

Web Dynpro 采用了一种高级的 MVC / Data Binding 架构模式，并且提供了非常友好的编程接口，Web 接口可以使用拖曳的方式进行开发。由于 Web Dynpro 运行于 SAP NetWeaver 平台，该平台在传统 Dynpro 开发上的优势也被使用得淋漓尽致。使用 Web Dynpro 可以非常迅速地开发出企业级的 Application 程序，而开发者只需要关注其业务流程，关于版本管理、质量控制、发布、性能等内容将由 NetWeaver 平台帮助完成。

1.1　ABAP

ABAP 作为一种面向特定应用的第四代编程语言最早在 20 世纪 80 年代被开发。它原本是作为一种报表语言应用在 SAP R/2 上，这是一个帮助大型公司在大型机上建立原材料管理和财务会计管理商务应用的平台。ABAP 本来也是德语 Allgemeiner Berichts Aufbereitungs Prozessor 的缩写，意思是通用报表预处理器。ABAP 第一次引入了“逻辑数据库”的概念，它在基本的数据库层提供了更高级的抽象。

虽然 SAP 1992 年就发布了 R/3，但 ABAP 一直用于为 R/3 系统编写程序。在 20 世纪 90 年代，随着计算机硬件的发展，越来越多的 SAP 的应用软件和系统都用 ABAP 来实现。直到 2001 年，几乎所有的基本功能都是由 ABAP 编程实现的。1999 年，SAP 在发布 R/3 4.6 版的同时也发布了一个对 ABAP 的面向对象扩展，叫作 ABAP Objects。

SAP 最新的开发平台 NetWeaver 同时支持 ABAP 和 Java。

2007 年，SAP 的应用服务器平台已经转向了 Web NetWeaver 的 SOA 平台，ABAP 也随即成为 SAP 中 Web Dynpro 的开发语言。

1.2　NetWeaver

NetWeaver 是 SAP 所有应用的技术集成平台的名字。SAP 的所有产品（包括 ECC（ERP Core Component）、CRM、SRM 和所有的解决方案（包括 mySAP ERP、mySAP Business Suite 等）都基于这一平台。

SAP NetWeaver 是一种可随时用于业务运作、面向服务的平台，适用于 SAP 的所有解决方案。SAP NetWeaver 平台内嵌了商务智能（BI）及无线射频识别（Radio Frequency Identification，RFID）功能，能够有效地进行主数据管理。同时，不同业务角色的用户可以通

过企业门户（Enterprise Portal，EP）上网访问企业内的各种信息。SAP NetWeaver 是近年来具有革命意义的基础应用和集成平台产品，它建立了面向服务的新的 SAP 企业服务信息系统基础架构（Eenterprise Service Architecture，ESA），提供了一种完全开放而且灵活的基础设施，使各公司能够通过现有的 IT 投资获取附加值，从而降低了企业的 IT 总成本。它既是应用平台，而且 DB 具有与 OS 无关性，通过与.NET 和 J2EE 的互动增加了灵活性；同时又实现了人员集成（多渠道访问、企业门户、协作），信息集成（商业智能、知识管理、主数据管理等）和流程集成（集成代理和商业流程管理等）。

Web Dynpro 就是基于这一平台，在 SAP 的各种解决方案中应运而生的。

1.3 Web Dynpro

Dynpro 通俗地讲就是屏幕主界面，其窗口一般包括标题栏、菜单栏、工具栏、状态栏和主界面。Dynpro 特指 SAP GUI 主界面区域的内容，是 ABAP 程序的一种。

Dynpro 是由用户界面元素和其背后的商业逻辑组成的动态程序。Dynpro 的主要组成部分如下：

1）属性，如当前的屏幕编号及下一个屏幕的编号等。

2）界面元素的位置，如文本标签、文本输入框、按钮等界面元素的位置安排。

3）界面元素的属性。

4）程序逻辑。这些程序将调用后台的 ABAP 代码模块。

1.3.1 Web Dynpro 的双重目的

1）尽可能避免对 UI 层进行编码。

2）允许业务应用程序独立于后台业务平台，以前端表现层的形式存在。

为了开发用户界面，Web Dynpro 提供了一个公开的元模型，这样就直接导致几乎无须编写程序代码。

从技术角度来讲，SAP 适用于 Java 和 ABAP 的 Web Dynpro 开发向基于 Web 的用户界面迈出了革命性的一步。它完全不同于 SAP 之前使用的任何设计范例，这标志着基于 Web 的企业资源规划（ERP）应用程序的开发有了重大突破。

Web Dynpro 应用程序是使用基于模型视图控制器（MVC）范例的声明式编程技术构建的。也就是说，用户可以指定客户端包括哪些用户界面元素，以及这些元素将从哪里获取数据。还可以通过声明的方式定义应用程序中可能的导航路径，用于创建用户界面的所有代码将在标准运行框架内自动生成。这可以减轻用户在编写 HTML 时的重复编码工作量，并使其能够与 JavaScript 进行交互。

SAP NetWeaver 7.0（SAP NetWeaver Application Server 7.0）及以上版本都可以使用 ABAP Web Dynpro。为了开发 Web Dynpro 应用程序的实体，对象导航器（事务代码 SE80）已有所强化。

Web Dynpro 专门用于支持结构化开发。其软件模块化单元是 Web Dynpro 组件，可组合起来构建复杂的应用程序。

1.3.2　Web Dynpro 技术要点

Web Dynpro 架构中包含 Web Dynpro 组件、UI 元素、窗口、视图，以及 Context。Web Dynpro 组件允许构建复杂的 Web 应用程序和开发可重用的、交互的实体，这可实现大段应用程序的嵌套。Web Dynpro 组件是与 UI 和 Web Dynpro 程序相关的其他实体的容器。与 UI 相关的实体是窗口和视图。视图的布局是客户端显示的页面的矩形部分（如浏览器）。视图包括 UI 元素，如输入字段和按钮。发送到客户端的整个页面可以设置为只包含一个视图，也可以是多个视图的组合。可能的视图组合及视图之间的数据流在窗口中定义。一个窗口可以包含任意数量的视图。一个视图可以嵌入到任意数量的窗口中。Web Dynpro 源代码位于 Web Dynpro 控制器中。控制器全局变量的层次存储被称为 Context。

除上述基本概念外，Web Dynpro 应用多与其他技术集成用以满足用户的需求；在 SAP NetWeaver 平台上，Web Dynpro 应用多以很多 Web Dynpro 高级概念来弥补 SAP 用户对 GUI 环境的依赖并满足用户的新需求。

第 2 章　集成 Adobe Flash Islands

Adobe Flash Islands 是基于 Adobe 的 Flex 技术。Adobe Flex 提供了一个通用的开发环境，可用于创建富互联网应用（RIA）。可以在所有主流浏览器平台和操作系统上开发并运行 RIA 程序。

Adobe Flex 创建的应用程序可以集成到的 Web Dynpro ABAP，这使得标准的 Web Dynpro 用新的组件得到加强。这些新的组件都实现了与 Adobe Flex 和相关的客户端/ PC 浏览器上本地运行。一个 Flash Islands UI 元素可以嵌入 Web Dynpro 的视图，可以访问 Web Dynpro 的 Context 节点，并且可以触发 Web Dynpro 的动作（Action）。

1．先决条件（Prerequisites）

要实现的 Adobe Flash Islands 的 Web Dynpro ABAP 应用程序，需要准备以下内容：

- 使用 Adobe 的 Flex Builder 2 或 3（开发环境）。
- 使用 Adobe Flash Player 9 Update 3（在客户端上使用 Adobe Flash Islands 控件）。
- 使用 SAP NetWeaver 所提供的 Adobe Flex 服务。该服务库（文件 WDIsland Library.swc 或 WDIslandLibrary-debug.swc）存储在 SAP/PUBLIC/BC/UR/nw7/FlashIslands 下的 MIME 处。
- 使用客户提供的 Adobe Flash 应用程序的 SWF 文件（连同相关的调试 SWF 文件）。

2．操作（Procedure）

在 Web Dynpro 中使用 Adobe Flash/Flex 应用程序。它必须被包含作为一个 Flash Island 的 UI 元素。这种包装可以使 Adobde Flash 应用程序与 Web Dynpro 框架之间进行通信。

2.1　Flash Island UI 元素的数据传输

Flash Island 的数据传输可以通过一个例子来描述。示例显示了一个条形图，该图创建于 Adobe Flex Builder 中，使用值从 SAP 系统中取得，学科（subject）和成绩（test）的值会显示在条形图上。在 Web Dynpro 应用程序中，最终用户可以将查询的成绩显示出来，若单击“总成绩”按钮，则 Adobe Flash 组件将取得 Web Dynpro 应用程序的总成绩及更新的条形图，运行结果如图 2-1 所示。

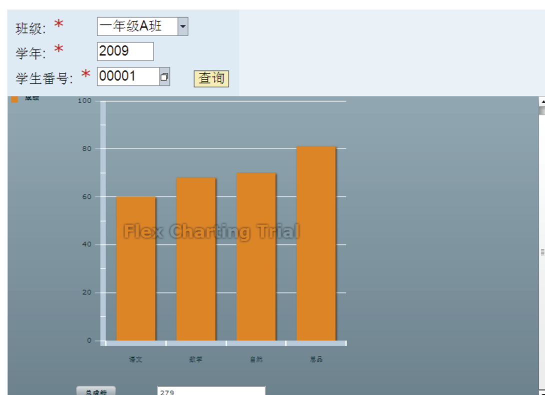


图 2-1

1．先决条件（Prerequisites）

该例的出发点是一个 SWF 文件（包括调试的 SWF 文件）及 SWF 文件的详细信息，如变量、列表和必要的动作。

注：Flex 文件区分大小写。

为了使 Adobe Flash 应用程序和 Web Dynpro 应用程序之间进行通信，需配置 Web Dynpro Island Library for Flex 到所需的项目：

- 这涉及 SWC 文件 WDIslandLibrary.swc 和 WDIslandLibrary-debug.swc。保存这两个文件到 MIME 资源库所在路径 SAP/PUBLIC/BC/UR/nw7/FlashIslands/下，以便可以随时访问它们。
- 将它们保存在 Adobe Flex Builder 下构建的路径。
- 在 Adobe Flex 编码需要时导入 Flash Islands 库（import sap.FlashIsland）。
- 为了数据传输能够正常进行，在 Adobe Flex 应用程序中使用 FlashIsland.register（<component_name>）语句注册相关的 Adobe Flex 组件。

2．操作（Procedure）

- 导入 SWF 文件及其调试的 SWF 文件。
- 在 Web Dynpro 应用程序中定义与 Adobe Flex 应用程序中相同名称的 Context 属性。
- 在调用 Flash Island 之前编辑 Context 节点，如供给函数。
- 编辑显示 Flash Island 控件的视图。

2.2 Flash Island UI 元素的事件

1．先决条件（Prerequisites）

在使用数据传输例子的 SWF 文件中，有一个附加的功能，可以单击按钮以改变在 Flash Island 值。

2．操作（Procedure）

在 Web Dynpro 应用程序中使用 Flash Island 的事件，意味着在 Flash Island 事件被触发时，这个事件把合计作为一个字符串参数传给 Web Dynpro 应用程序。该事件称为 onClickButton。

1）在 Web Dynpro 应用程序集成的 Flash Island UI 元素中添加一个 GACEvent。

2）GACEven 元素必须设定以下属性值。

- name：onClickButton。
- OnAction：相关的 Web Dynpro 动作。

3）设定 GACEvent 参数 GACEventParameter 元素的名称属性及其类型属性匹配预期的类型。

4）WDEvent 使用动作处理程序方法得到参数在 Web Dynpro 应用中做相关处理。

2.3 编辑 Adobe Flex 应用程序

1）启动 Adobe Flex 编辑器，在 Flex 工程编辑界面创建 Flex 工程，如图 2-2 所示。

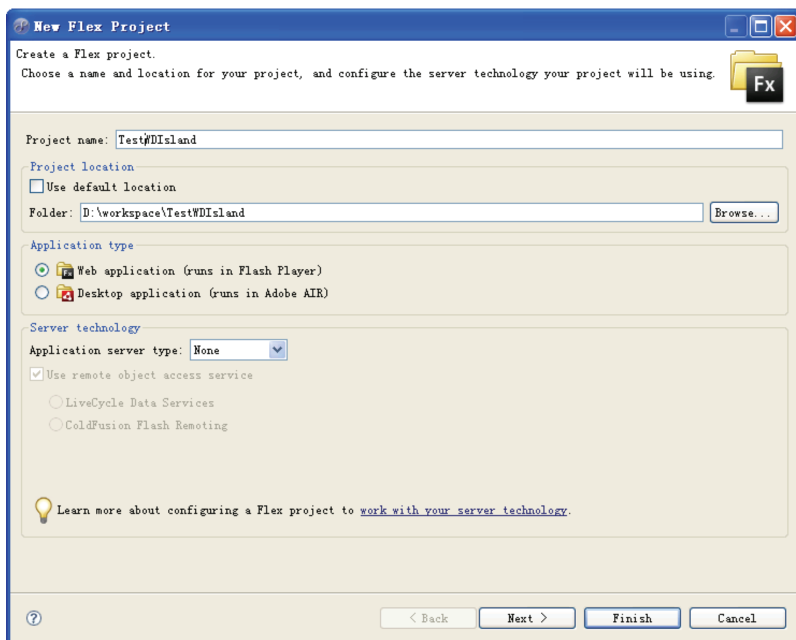


图 2-2

单击 Next > 按钮，显示编译路径设定对话框，如图 2-3 所示。

设定编译文件路径并单击 Next > 按钮，显示编译路径设定对话框（文件及集成环境），如图 2-4 所示。

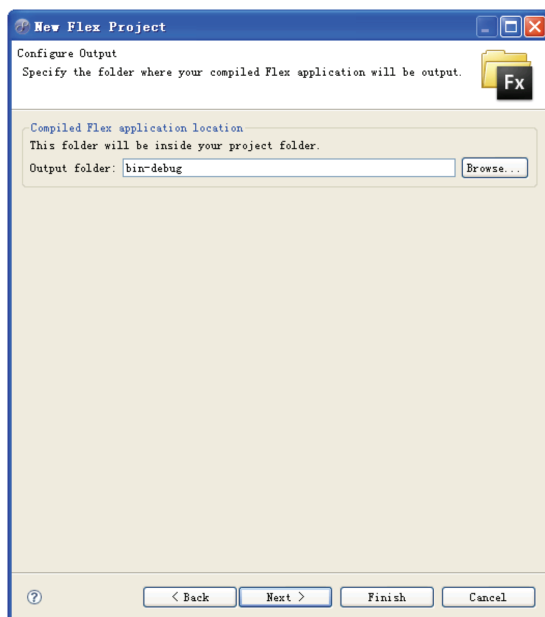


图 2-3

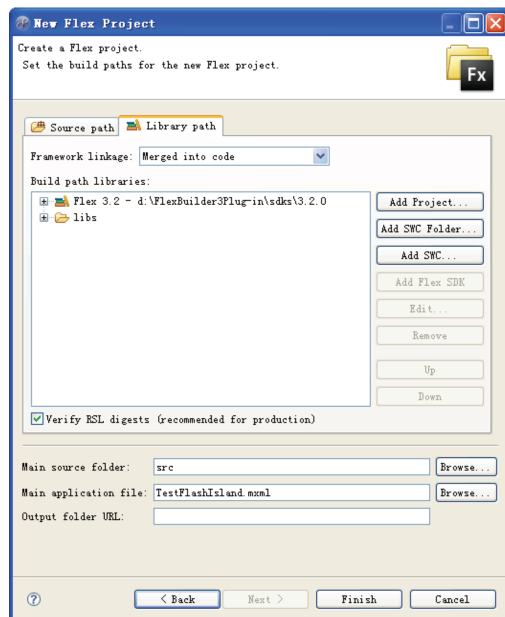


图 2-4

单击 Add SWC... 按钮，添加编译所需文件 WDIslandLibrary.swc，如图 2-5 所示。

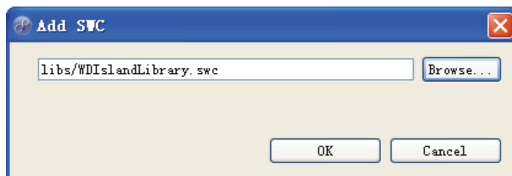


图 2-5

单击 OK 按钮再单击 Finish 按钮。

2）在 Adobe Flex 编辑器中的工程属性编辑界面更改 Flex 工程属性，如图 2-6 所示。

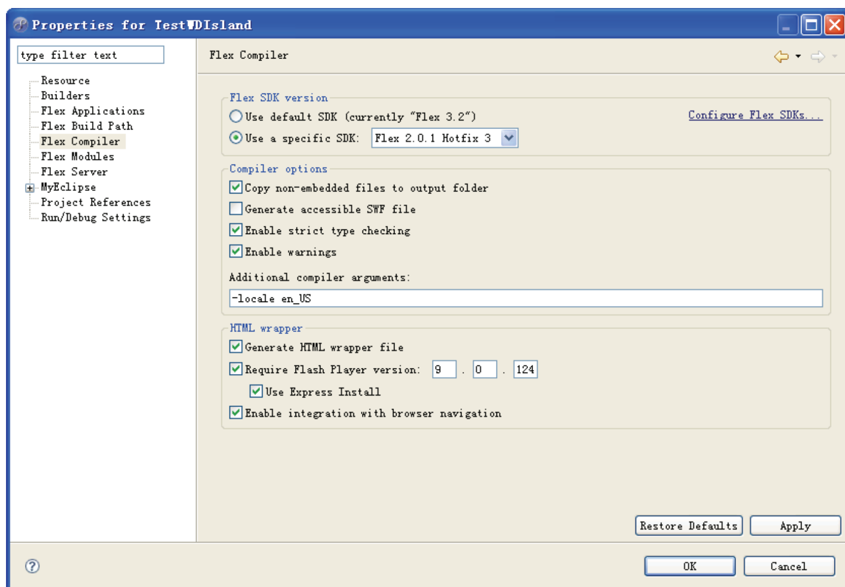


图 2-6

3）在 Adobe Flex 编辑器中的 Flex 应用文件编辑界面，编辑并编译 Flex 应用文件如下：

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application  xmlns:mx="http://www.adobe.com/2006/mxml"  layout="absolute"  initialize=
"init()" xmlns:comp="components.*">
    <mx:Script>
        <![CDATA[
        import sap.FlashIsland;
        import mx.collections.ArrayCollection;
        [Bindable] public var columnTest:ArrayCollection ;
        [Bindable] public var test:String ;
        [Bindable] public var tittle:String ;
        [Bindable] public var subjects:String ;
        [Bindable] public var sum:String    ;
         public function init():void {
                FlashIsland.register(this);
                }
         private function doSum(event:Event):void
         {
```

```
            //定义参数
        var param:Object = {param1:'Value 1', param2:'Value 2'};
        var param1:String ;
            //求和
       for(var i:int = 0;i<columnTest.length;i++)
       {
       var obj:int = int(columnTest.getItemAt(i)[test]);
       var obj1: int = obj1 + obj;
       }
        param1 =String(obj1);//.test;
        FlashIsland.fireEvent(this, "onClickButton", {param:param1} );
        }

       ]]>
    </mx:Script>

       <mx:ColumnChart x="101" y="10" id="column" dataProvider="{columnTest}"showDataTips=" true">

       <mx:verticalAxis>
           <mx:LinearAxis maximum="100"/>
       </mx:verticalAxis>
       <mx:horizontalAxis>
        <mx:CategoryAxis id = "columncate" categoryField="{subjects}" dataProvider="{columnTest}" />
        </mx:horizontalAxis>
            <mx:series >
                <mx:ColumnSeries displayName="{tittle}" yField="{test}"dataProvider="{columnTest}"/>
            </mx:series>
       </mx:ColumnChart>
       <mx:Legend dataProvider="{column}"/>
       <mx:Button x="101" y="440" label="总成绩" click="doSum(event)"/>
       <mx:TextInput x="221" y="440" editable="false" text="{sum}"/>

    </mx:Application>
```

注：sum 这个变量应该作为绑定变量从 Flash Island 传递到 Web Dynpro 应用程序中，test、tittle、subjects 则相反。

自动生成编译文件如图 2-7 所示。

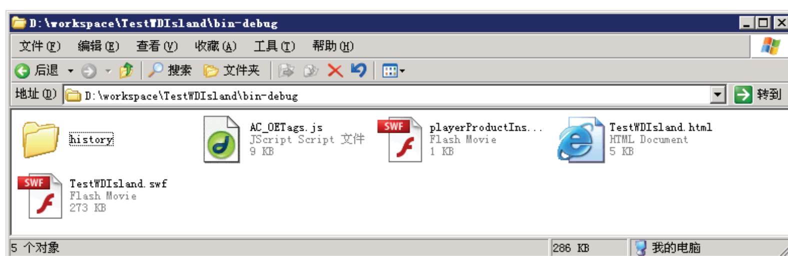


图 2-7

2.4 编辑 Web Dynpro 应用程序

1）启动 ABAP 工作台，在 Web Dynpro 组件编辑界面，创建 Web Dynpro 组件。

在下拉列表中选中“Web Dynpro Comp./Intf”选项，在相应的文本框中输入程序的名称，如图 2-8 所示。

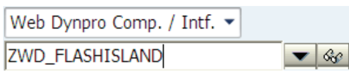


图 2-8

单击图 2-8 中的按钮，弹出如图 2-9 所示的对话框。



图 2-9

单击图 2-9 中的 是 按钮，弹出组件编辑对话框，编辑组件相关信息，如图 2-10 所示。

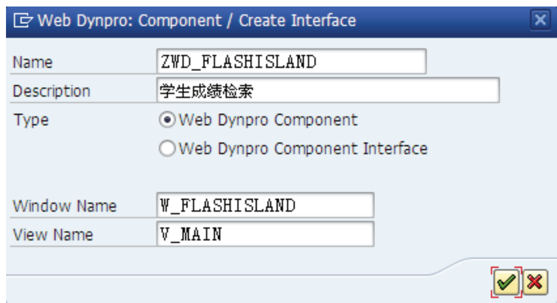


图 2-10

单击图 2-10 中的按钮，并添加新视图，如图 2-11 所示。

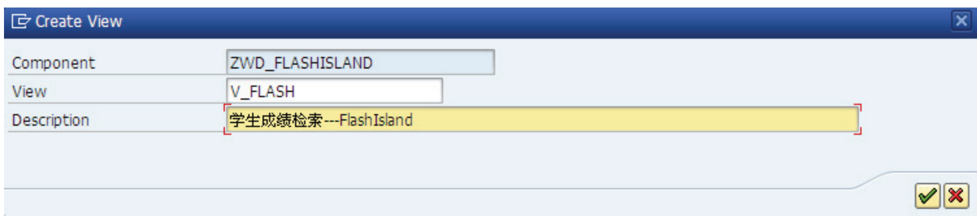


图 2-11

为组件中的相关对象编写适当的Description（注释），如图 2-12 所示。

Object Name	Description
ZWD_FLASHISLAND	学生成绩检索
COMPONENTCONTROLL	学生成绩检索---组件控制器
Component Interface	
INTERFACECONTROL	学生成绩检索---组件控制器
Interface Views	
W_FLASHISLAND	学生成绩检索---窗体
Views	
V_FLASH	学生成绩检索---FlashIsland
V_MAIN	学生成绩检索---视图
Windows	
W_FLASHISLAND	学生成绩检索---窗体

图 2-12

2）在 ABAP 工作台中的 Web Dynpro 组件编辑界面导入 Flex 编译文件。

在组件上单击鼠标右键，在弹出的菜单中单击“Create”→“Mime Object”→“Import”命令，导入文件（见图 2-13），弹出如图 2-14 所示的对话框。

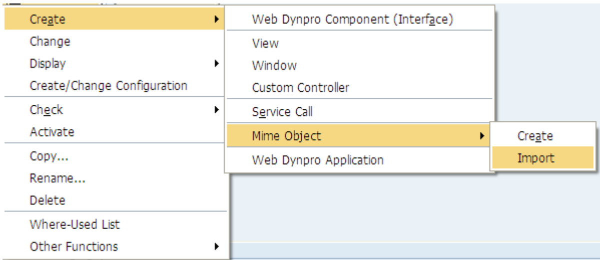


图 2-13



图 2-14

选中编译完成的 Flex 文件，单击💾按钮。

3）在 ABAP 工作台中的组件编辑界面编辑组件控制器属性。

在左侧程序的树形结构菜单上双击组件控制器，Attributes 编辑如图 2-15 所示。

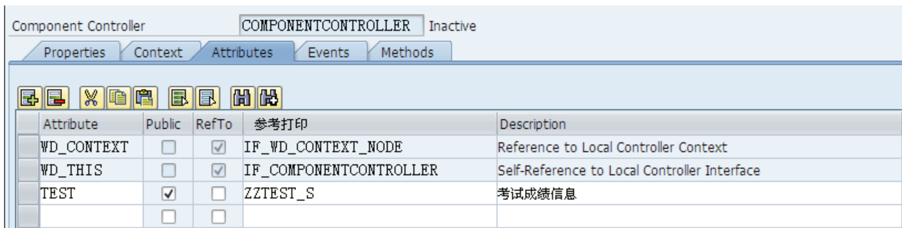


图 2-15

4）在 ABAP 工作台中的视图编辑界面编辑视图 V_MAIN。

在左侧程序的树形结构菜单上双击视图 V_MAIN，选中“Context”选项卡，编辑结果如图 2-16 所示。

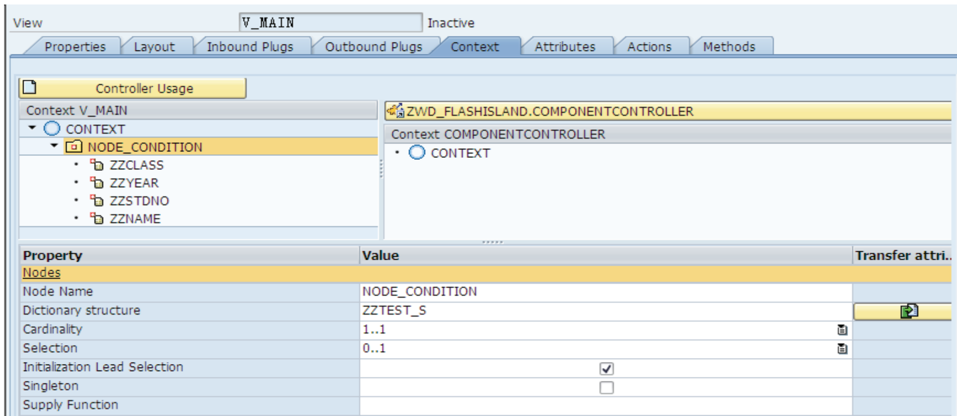


图 2-16

选中“Outbound Plugs”选项卡，编辑出站插头如图 2-17 所示。

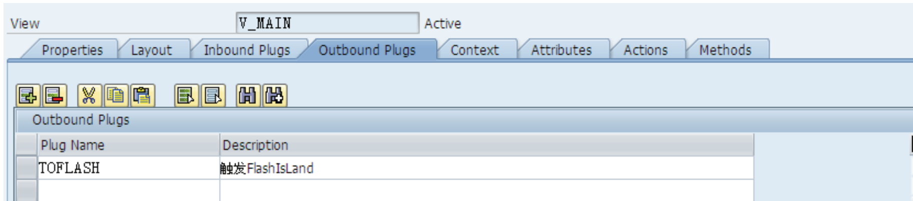


图 2-17

选中“Layout”选项卡，编辑查询条件信息，并添加视图容器 UI 元素如图 2-18 所示。

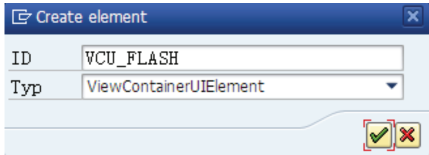


图 2-18

编辑结果如图 2–19 所示。

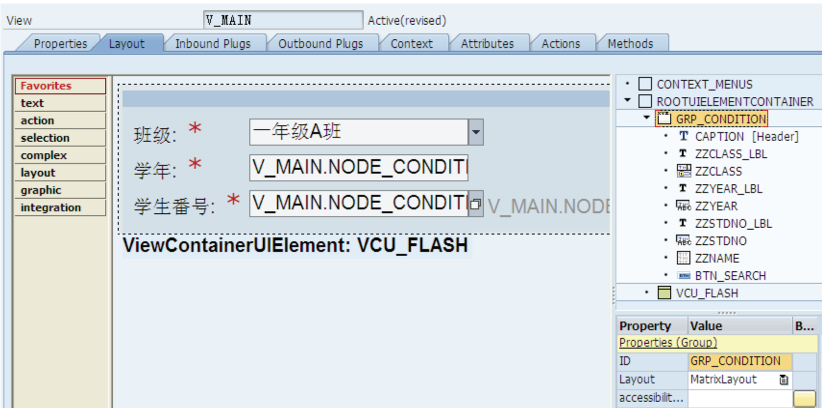


图 2–19

为“BTN_SEARCH”按钮编辑动作 GET_TEST 如图 2–20 所示。

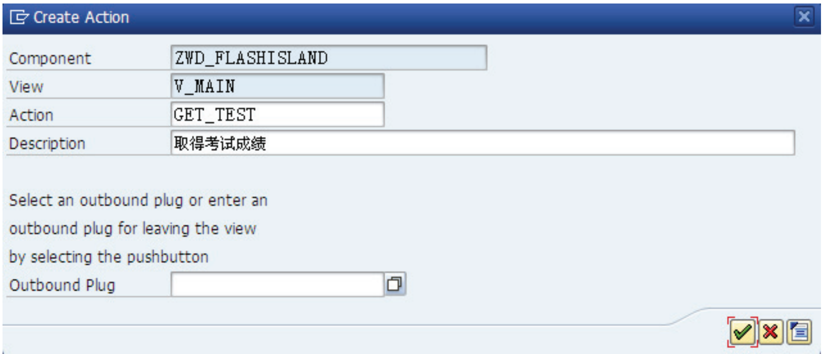


图 2–20

动作 GET_TEST 处理方法编辑如下：

```
METHOD ONACTIONGET_TEST .
    DATA LO_ND_NODE_CONDITION TYPE REF TO IF_WD_CONTEXT_NODE.

    DATA LO_EL_NODE_CONDITION TYPE REF TO IF_WD_CONTEXT_ELEMENT.
    DATA LS_NODE_CONDITION TYPE WD_THIS->ELEMENT_NODE_CONDITION.

*     navigate from <CONTEXT> to <NODE_CONDITION> via lead selection
    LO_ND_NODE_CONDITION = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->
    WDCTX_NODE_CONDITION ).

*    get element via lead selection
    LO_EL_NODE_CONDITION = LO_ND_NODE_CONDITION->GET_ELEMENT( ).

*    get all declared attributes
```

```
        LO_EL_NODE_CONDITION->GET_STATIC_ATTRIBUTES(
         IMPORTING
          STATIC_ATTRIBUTES = LS_NODE_CONDITION ).
        * 取得成绩信息
          SELECT  SINGLE     *
                FROM  ZZTEST_V
                INTO CORRESPONDING FIELDS OF  WD_COMP_CONTROLLER->TEST
                WHERE  ZZCLASS  = LS_NODE_CONDITION-ZZCLASS
                AND    ZZYEAR   = LS_NODE_CONDITION-ZZYEAR
               AND    ZZSTDNO  = LS_NODE_CONDITION-ZZSTDNO.

      *         set single attribute
        LO_EL_NODE_CONDITION->SET_ATTRIBUTE(
          NAME =  `ZZNAME`
          VALUE = LS_NODE_CONDITION-ZZNAME ).

    WD_THIS->FIRE_TOFLASH_PLG(
    ).

  ENDMETHOD.
```

5）在 ABAP 工作台中的视图编辑界面编辑视图 V_FLASH。

在左侧程序的树形结构菜单上双击视图 V_FLASH，选中“Context”选项卡，编辑结果如图 2-21 所示。

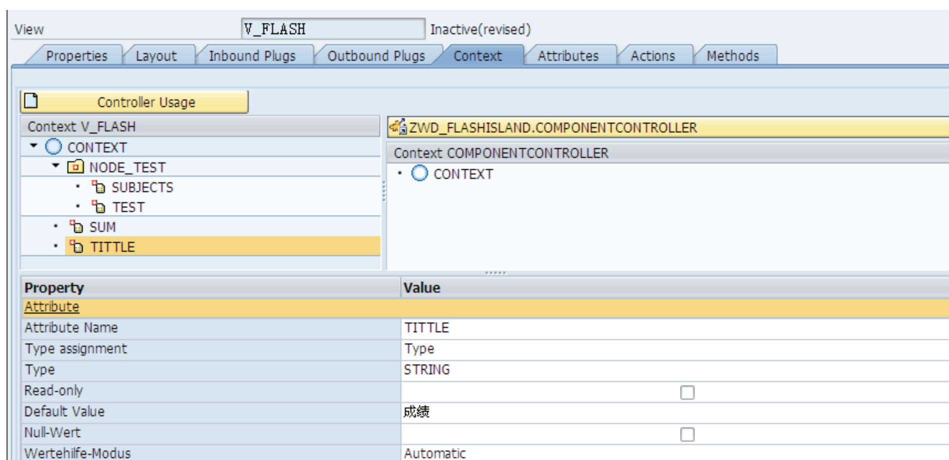


图 2-21

注：在 Context 中定义的变量要和在 FlashIsland 定义的变量一致，否则变量值无法传递。

选中“Inbound Plugs”选项卡，编辑入站插头如图 2-22 所示。

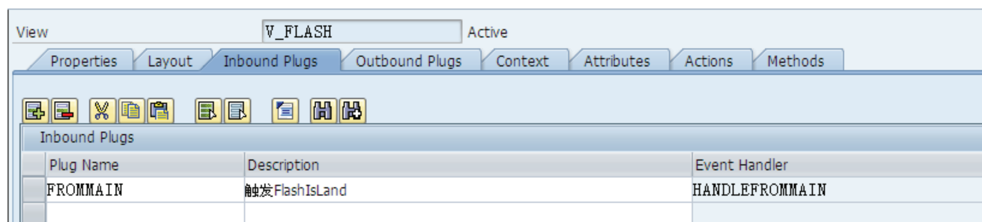


图 2-22

选中“Layout”选项卡，选中右侧的根节点“ROOTUIELEMENTCONTAINER”，单击鼠标右键，按照图 2-23 所示的路径转换 UI 元素类型。

选中变换属性为“FlashIsland”，弹出如图 2-24 所示的对话框。

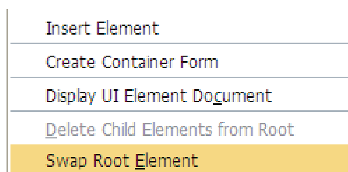


图 2-23

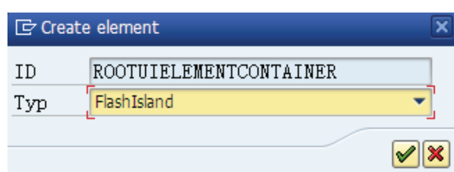


图 2-24

变换后属性编辑如图 2-25 所示。

Property	Value
Properties (FlashIsland)	
ID	ROOTUIELEMENTCONTAINER
enabled	☑
height	450px
swfFile	{ZWD_FLASHISLAND}/TestWDIsland.swf
tooltip	
visible	Visible
width	800px

图 2-25

选中右侧的根目录“ROOTUIELEMENTCONTAINER”，单击鼠标右键，按照图 2-26 所示的路径创建 UI 元素 GACDATASOURCE、GACEVENT 和 GACPROPERTY。

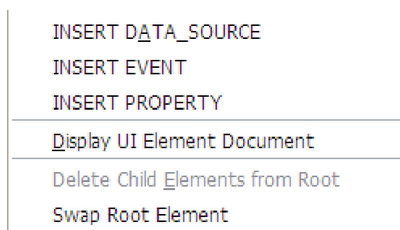


图 2-26

选中创建的 UI 元素“GACEVENT [Events]”单击鼠标右键，按照图 2-27 所示的路径创建事件的参数。

编辑结果如图 2-28 所示。

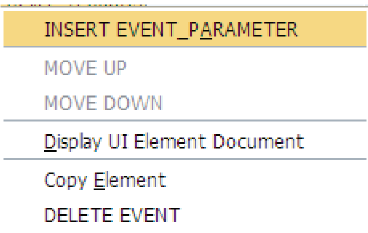


图 2-27

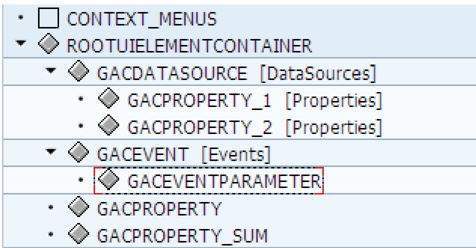


图 2-28

注：GACDataSource 用于绑定 Context 节点，与 Flex 文件中的 columnTest 相对应。GACProperties 用于绑定 Context 节点中的属性，对应 Flex 文件中的变量。GACEVENT 用于绑定视图中的 Action，对应 Flex 文件中的事件 onClickButton。实例中相关的属性设定见表 2-1。

表 2-1

Property	Value	Bi...
Properties (GACDataSource)		
ID	GACDATASOURCE	
dataSource	V_FLASH.NODE_TEST	
name	columnTest	

Property	Value	Bi...
Properties (GACProperty)		
ID	GACPROPERTY_1	
name	subjects	
readOnly	☑	
value	V_FLASH.NODE_TE...	

Property	Value	Bi...
Properties (GACProperty)		
ID	GACPROPERTY_2	
name	test	
readOnly	☑	
value	V_FLASH.NODE_TE...	

Property	Value	Bi...
Properties (GACEvent)		
ID	GACEVENT	
name	onClickButton	
Events		
onAction	GET_SUM	

Property	Value	Bi...
Properties (GACEventParameter)		
ID	GACEVENTPARAME...	
name	param	
type	02	

Property	Value	Bi...
Properties (GACProperty)		
ID	GACPROPERTY	
name	tittle	
readOnly	☑	
value	V_FLASH.TITTLE	

Property	Value	Bi...
Properties (GACProperty)		
ID	GACPROPERTY_SUM	
name	sum	
readOnly	☑	
value	V_FLASH.SUM	

选中“Action”选项卡，新建动作 GET_NUM 如图 2-29 所示。



图 2-29

编辑结果如图 2-30 所示。

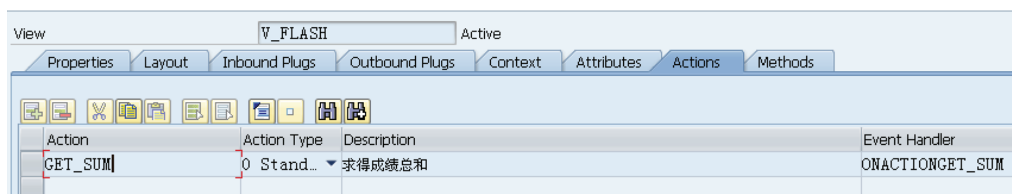


图 2-30

动作对应事件处理程序编辑如下：

```
METHOD ONACTIONGET_SUM .

    DATA LO_EL_CONTEXT TYPE REF TO IF_WD_CONTEXT_ELEMENT.
    DATA LS_CONTEXT TYPE WD_THIS->ELEMENT_CONTEXT.
    DATA LV_SUM TYPE WD_THIS->ELEMENT_CONTEXT-SUM.

*    get element via lead selection
 LO_EL_CONTEXT = WD_CONTEXT->GET_ELEMENT( ).

*  @TODO fill attribute
 WDEVENT->GET_DATA(
   EXPORTING
      NAME   = 'param'
   IMPORTING
      VALUE  = LV_SUM
          ).

*    set single attribute
 LO_EL_CONTEXT->SET_ATTRIBUTE(
    NAME =  `SUM`
    VALUE = LV_SUM ).
ENDMETHOD.
```

选中“Methods”选项卡，入站插头 FROMMAIN 处理程序如下：

```
method HANDLEFROMMAIN .
   DATA LO_ND_NODE_TEST TYPE REF TO IF_WD_CONTEXT_NODE.

   DATA LS_NODE_TEST TYPE WD_THIS->ELEMENT_NODE_TEST.
   DATA LT_NODE_TEST TYPE WD_THIS->ELEMENTS_NODE_TEST.

*    navigate from <CONTEXT> to <NODE_TEST> via lead selection
   LO_ND_NODE_TEST=WD_CONTEXT->GET_CHILD_NODE(NAME=WD_THIS->WDCTX_NODE_TEST)
LO_ND_NODE_TEST->INVALIDATE( ).
*   @TODO compute values
   MOVE '语文' TO  LS_NODE_TEST-SUBJECTS.
   MOVE WD_COMP_CONTROLLER->TEST-ZZTENKOK TO  LS_NODE_TEST-TEST.
```

```
        APPEND LS_NODE_TEST TO LT_NODE_TEST.
        MOVE '数学' TO  LS_NODE_TEST-SUBJECTS.
        MOVE WD_COMP_CONTROLLER->TEST-ZZTENSAN TO  LS_NODE_TEST-TEST.
        APPEND LS_NODE_TEST TO LT_NODE_TEST.
        MOVE '自然' TO  LS_NODE_TEST-SUBJECTS.
        MOVE WD_COMP_CONTROLLER->TEST-ZZTENRIK TO  LS_NODE_TEST-TEST.
        APPEND LS_NODE_TEST TO LT_NODE_TEST.
        MOVE '思品' TO  LS_NODE_TEST-SUBJECTS.
        MOVE WD_COMP_CONTROLLER->TEST-ZZTENSHA TO  LS_NODE_TEST-TEST.
      APPEND LS_NODE_TEST TO LT_NODE_TEST.
      *
         LO_ND_NODE_TEST->BIND_TABLE( NEW_ITEMS = LT_NODE_TEST SET_INITIAL_ELEME
NTS = ABAP_FALSE ).
      endmethod.
```

6）在 ABAP 工作台中的窗体编辑界面编辑视图间导航链接。

在左侧程序的树形结构菜单上双击窗体 W_FLASHISLAND，选中“Window”选项卡，编辑导航如图 2-31 所示。

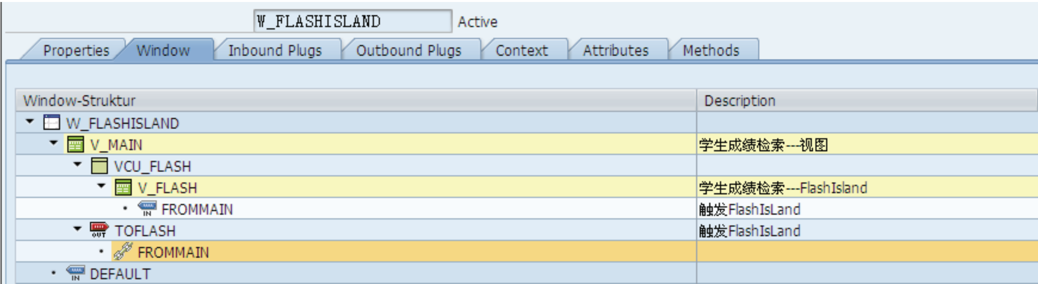


图 2-31

7）在 ABAP 工作台中的 Web Dynpro Application 编辑界面创建 Web Dynpro Application，如图 2-32 所示。

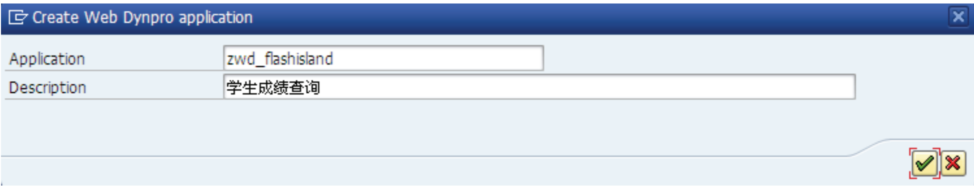


图 2-32

第 3 章　文 件 操 作

Web Dynpro 应用程序可以使用 FileDownload 控件导出的文件，从而实现通常的安全要求。该下载的功能在任何一个操作处理程序中都是可用的。例如，当单击一个的按钮，LinkToAction 或（作为正常的程序设计不应该这样做）切换选项卡 TabStrip 时或选中一个单选按钮，都可以进行下载。当用户单击 FileDownload 的 UI 元素时，将产生二进制数据流。对于这个数据流，会产生相应的 URL 的结果并显示在一个单独的浏览器窗体中。

（1）特点（Features）

无论是导出一个文件或多个文件，都以相同的方式显示数据并存储文件。

① 在同一窗口中显示，如图 3-1 所示。



图 3-1

② 在同一窗口中保存查询，如图 3-2 所示。

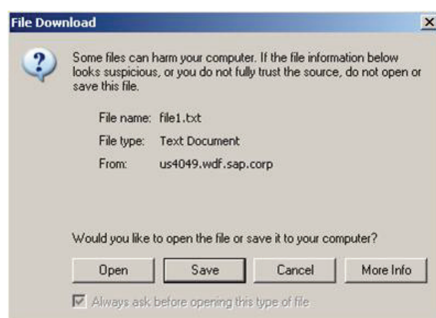


图 3-2

③ 在一个平行的窗体中显示。

④ 在一个平行的窗体中保存查询。

注：使用 Internet Explorer 时，多个文件的保存查询需通过外部窗体。因此，同时下载多个文件都是通过外部窗体。

（2）操作（Procedure）

使用类 CL_WD_RUNTIME_SERVICES 中的方法 ATTACH_FILE_TO_RESPONSE 实现文件导出。

下面介绍编辑 Web Dynpro 应用程序的方法。

该例以学生相片为例，将学生相片所在文件上传到服务器，并通过链接提供下载或显示

在浏览器中。

1）启动 ABAP 工作台，在 Web Dynpro 组件编辑界面创建组件，如图 3-3 所示。

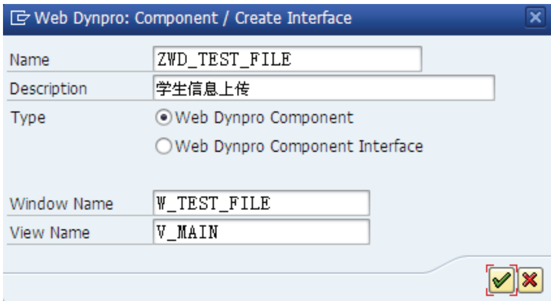


图 3-3

编辑结果如图 3-4 所示。

Object Name	Description
ZWD_TEST_FILE	学生信息上传
COMPONENTCONTROLL	学生信息上传---组件控制器
Component Interface	
INTERFACECONTROL	学生信息上传---组件控制器
Interface Views	
W_TEST_FILE	学生信息上传---窗体
Views	
V_MAIN	学生信息上传---视图
Windows	
W_TEST_FILE	学生信息上传---窗体

图 3-4

2）在 ABAP 工作台中的视图编辑界面编辑视图。

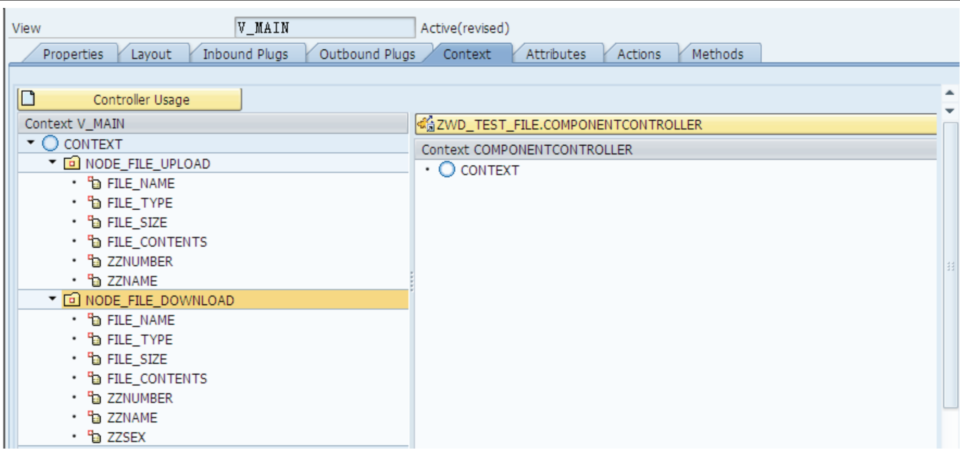
在左侧的树形结构菜单上双击视图，选择“Context”选项卡，添加 Context 节点，见表 3-1。

表 3-1

NODE_FILE_UPLOAD	NODE_FILE_DOWNLOAD
Create Node Node Name: NODE_FILE_UPLOAD Interface Node: No Input Element (Ext.): No Dictionary structure: Cardinality: 1..1 Selection: 0..1 Init. Lead Selection: Yes Singleton: No Supply Function: Add Attribute from Structure / Additional Node	Create Node Node Name: NODE_FILE_DOWNLOAD Interface Node: No Input Element (Ext.): No Dictionary structure: Cardinality: 0..n Selection: 0..1 Init. Lead Selection: Yes Singleton: No Supply Function: Add Attribute from Structure / Additional Node
基数：1.1	基数：0.n

（续）

属　　性	类　　型	属　　性	类　　型
FILE_NAME	STRING	FILE_NAME	STRING
FILE_TYPE	STRING	FILE_TYPE	STRING
FILE_SIZE	STRING	FILE_SIZE	STRING
FILE_CONTENTS	XSTRING	FILE_CONTENTS	XSTRING
编辑结果			



选择“Layout”选项卡，添加组 UI 元素，如图 3-5 所示。

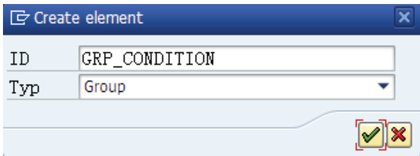


图 3-5

在组 UI 元素下创建表单如图 3-6 所示。

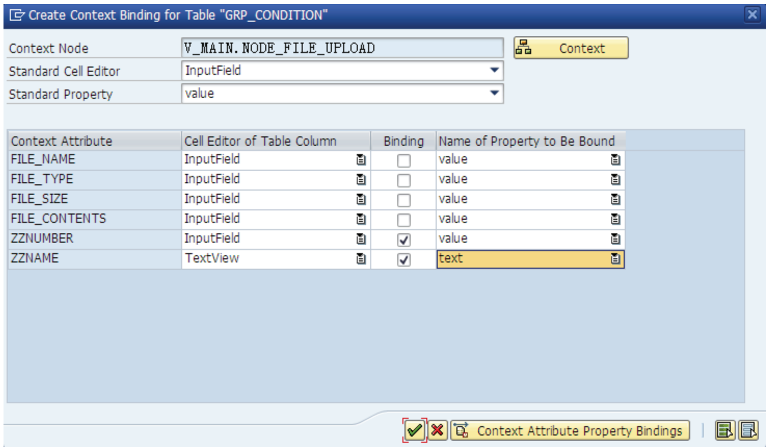


图 3-6

添加文件上传 UI 元素如图 3-7 所示。

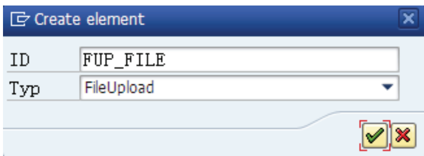


图 3-7

属性设定如图 3-8 所示。

Property	Value	B...
Properties (FileUpload)		
ID	FUP_FILE	
Layout Data	MatrixData	
activateAccessKey	☐	
contextMenuBehavi...	Inherit	
contextMenuId		
data	V_MAIN.NODE_FILE_UPLOAD.FILE_CONTENTS	
enabled	☑	
explanation		
fileName	V_MAIN.NODE_FILE_UPLOAD.FILE_NAME	
mimeType	V_MAIN.NODE_FILE_UPLOAD.FILE_TYPE	
state	Normal Item	
textDirection	Inherit	
tooltip		
visible	Visible	
width		

图 3-8

添加按钮 UI 元素如图 3-9 所示。

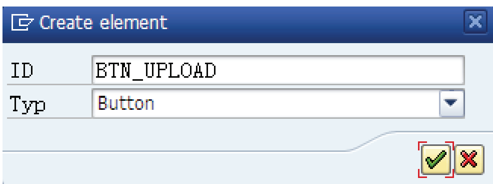


图 3-9

分配 Action 如图 3-10 所示。

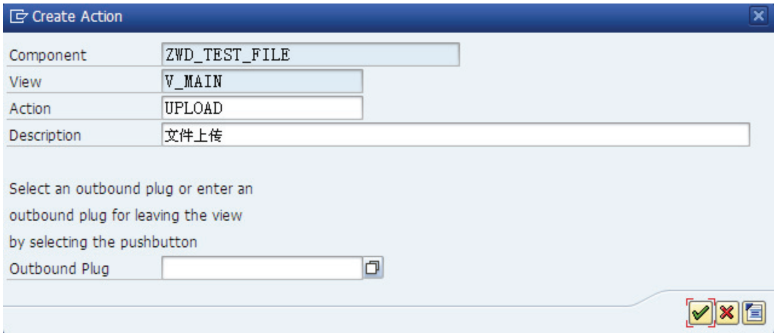


图 3-10

动作 Action 对应处理程序编辑如下：

```
METHOD ONACTIONUPLOAD .
* Node，Structure，Table 定义
  DATA LO_ND_NODE_FILE_UPLOAD TYPE REF TO IF_WD_CONTEXT_NODE.

  DATA LO_EL_NODE_FILE_UPLOAD TYPE REF TO IF_WD_CONTEXT_ELEMENT.
  DATA LS_NODE_FILE_UPLOAD TYPE WD_THIS->ELEMENT_NODE_FILE_UPLOAD.

  DATA LO_ND_NODE_FILE_DOWNLOAD TYPE REF TO IF_WD_CONTEXT_NODE.

  DATA LS_NODE_FILE_DOWNLOAD TYPE WD_THIS->ELEMENT_NODE_FILE_DOWNLOAD.
  DATA LT_NODE_FILE_DOWNLOAD TYPE WD_THIS->ELEMENTS_NODE_FILE_DOWNLOAD.

*     navigate from <CONTEXT> to <NODE_FILE_UPLOAD> via lead selection
  LO_ND_NODE_FILE_UPLOAD = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THI
S->WDCTX_NODE_FILE_UPLOAD ).
*     get element via lead selection
  LO_EL_NODE_FILE_UPLOAD = LO_ND_NODE_FILE_UPLOAD->GET_ELEMENT( ).
*     get all declared attributes
  LO_EL_NODE_FILE_UPLOAD->GET_STATIC_ATTRIBUTES(
     IMPORTING
        STATIC_ATTRIBUTES = LS_NODE_FILE_UPLOAD ).
* 由 Condition 到 List 传值
 MOVE-CORRESPONDING LS_NODE_FILE_UPLOAD  TO LS_NODE_FILE_DOWNLOAD .
 SELECT SINGLE
                ZZNUMBER
                ZZNAME
                ZZSEX
    FROM  ZMNUMBER_T
                INTO (LS_NODE_FILE_DOWNLOAD-ZZNUMBER,
                      LS_NODE_FILE_DOWNLOAD-ZZNAME ,
                      LS_NODE_FILE_DOWNLOAD-ZZSEX )
          WHERE  ZZNUMBER  = LS_NODE_FILE_UPLOAD-ZZNUMBER .

*      navigate from <CONTEXT> to <NODE_FILE_DOWNLOAD> via lead selection
  LO_ND_NODE_FILE_DOWNLOAD = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->
WDCTX_NODE_FILE_DOWNLOAD ).
*  *     @TODO compute values
  LS_NODE_FILE_DOWNLOAD-FILE_SIZE = XSTRLEN( LS_NODE_FILE_UPLOAD-FILE_CONTENTS ).

  APPEND LS_NODE_FILE_DOWNLOAD TO   LT_NODE_FILE_DOWNLOAD.
*
   LO_ND_NODE_FILE_DOWNLOAD->BIND_TABLE( NEW_ITEMS = LT_NODE_FILE_DO
WNLOAD SET_INITIAL_ELEMENTS = ABAP_FALSE ).
ENDMETHOD.
```

添加托盘UI元素如图3-11所示。

在托盘 UI 元素下添加表 UI 元素如图 3-12 所示。

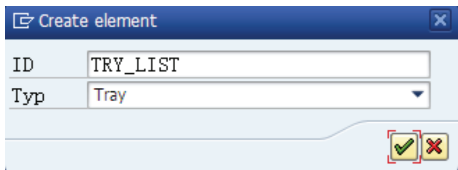


图 3-11

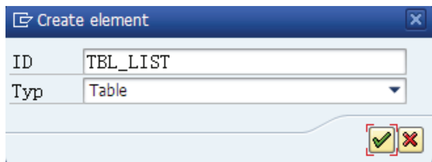
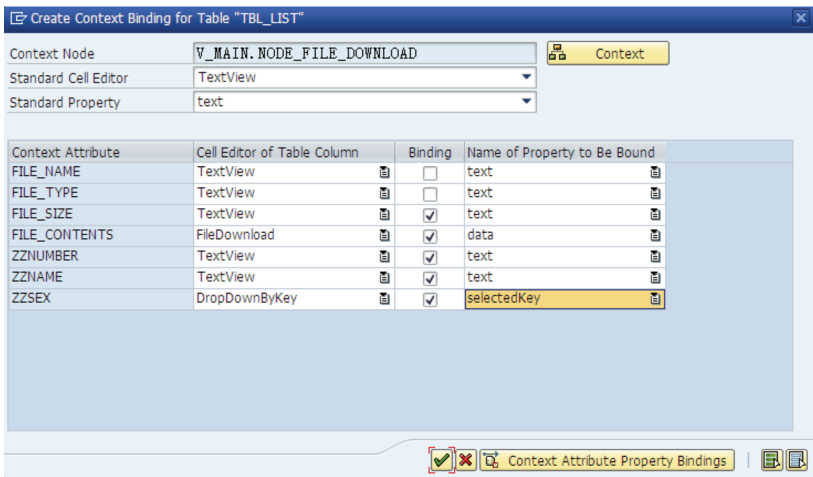


图 3-12

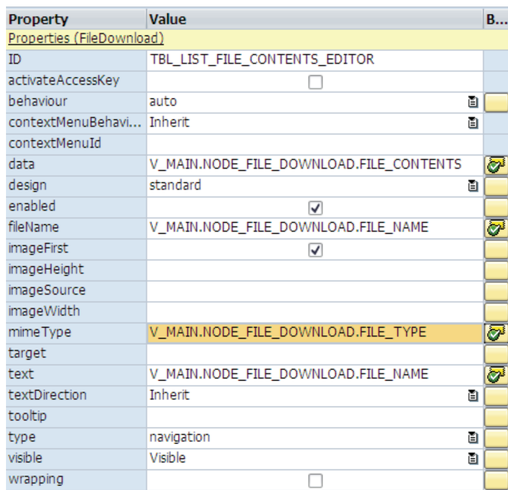
为表 UI 元素创建绑定如图 3-13 所示。



Context Attribute	Cell Editor of Table Column	Binding	Name of Property to Be Bound
FILE_NAME	TextView	☐	text
FILE_TYPE	TextView	☐	text
FILE_SIZE	TextView	☑	text
FILE_CONTENTS	FileDownload	☑	data
ZZNUMBER	TextView	☑	text
ZZNAME	TextView	☑	text
ZZSEX	DropDownByKey	☑	selectedKey

图 3-13

其中绑定的 FileDownload 元素属性绑定如图 3-14 所示。



Property	Value	B...
Properties (FileDownload)		
ID	TBL_LIST_FILE_CONTENTS_EDITOR	
activateAccessKey	☐	
behaviour	auto	
contextMenuBehavi...	Inherit	
contextMenuId		
data	V_MAIN.NODE_FILE_DOWNLOAD.FILE_CONTENTS	
design	standard	
enabled	☑	
fileName	V_MAIN.NODE_FILE_DOWNLOAD.FILE_NAME	
imageFirst	☑	
imageHeight		
imageSource		
imageWidth		
mimeType	V_MAIN.NODE_FILE_DOWNLOAD.FILE_TYPE	
target		
text	V_MAIN.NODE_FILE_DOWNLOAD.FILE_NAME	
textDirection	Inherit	
tooltip		
type	navigation	
visible	Visible	
wrapping	☐	

图 3-14

编辑结果如图 3-15 所示。



图 3-15

3）在 ABAP 工作台中的 Web Dynpro Application 编辑界面创建 Web Dynpro Application。编辑结果如图 3-16 所示。

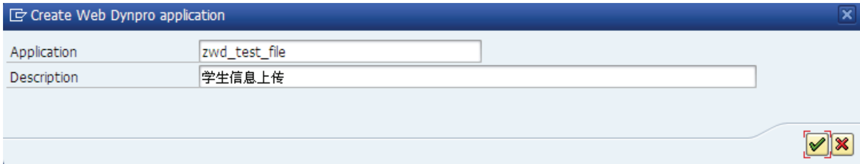


图 3-16

4）在 ABAP 工作台中运行 Web Dynpro Application。
运行结果如图 3-17 所示。

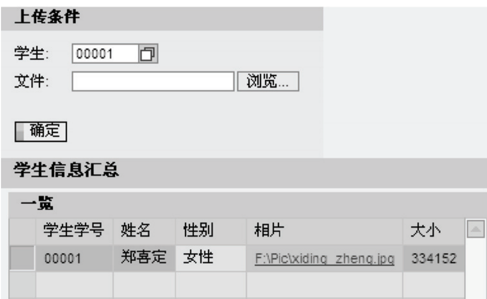


图 3-17

单击文件链接，如图 3-18 所示。

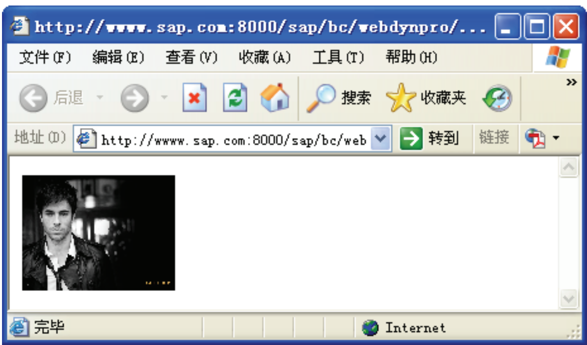


图 3-18

第 4 章　门户集成（Portal Integration）

Web Dynpro ABAP 应用程序可以被集成到 SAP 企业门户（SAP　Enterprise Portal）。也就是说，它们可以绑定到一个门户网站导航的 iView。

这种把一个 Web Dynpro 应用程序绑定到一个门户网站应用程序的编程方法，适合约束单独的应用程序的开发和应用，偏重于测试目的。

此外，它可以解决 Web Dynpro 应用程序内的门户功能。因此，程序员可以访问门户网站管理方法（接口 IF_WD_PORTAL_INTEGRATION），作为不同功能的源代码模板，调用 Web Dynpro 代码向导。这些包括以下内容：

1）使用门户网站事件。

2）在门户网站内导航任何 Web Dynpro 应用。它支持下面的导航类型：

- 基于对象的导航。
- 绝对导航。
- 相对导航。

3）使用 WorkProtect 模式。

4.1　绑定到门户的准备

要将 Web Dynpro 应用程序集成到门户，必须将 Web Dynpro 应用程序所在系统集成到门户系统，实现单点登录（Single-Sign-On），需满足以下先决条件：

1）在门户系统中配置一个用户，并且在门户中被分配到一个合适的角色（如内容管理角色）。该角色需包含所有必要的授权和工具使用。

2）Web Dynpro 应用程序所在的 ABAP 系统对于门户必须是已知的。这一点是由于进入门户系统访问数据需要特殊授权，必要的话也要能够进入 ABAP 系统。在接下来的步骤中，将需要 ABAP 系统别名，该别名需要被分配到门户网站。

3）接下来要测试应用程序，程序员必须也有一个 ABAP 系统用户。为了避免调用该应用程序时重新登录 ABAP 系统，使用用户映射，可以连接门户网站与 ABAP 的系统。

一旦这些技术的先决条件得到满足，登录到门户网站，便可以在最初的导航屏幕功能内选择内容管理。

注：具体如何实施基于 SAP NetWeaver EP 的单点登录请参阅相关文档。本例中将 Web Dynpro 应用程序所在系统集成到门户系统（xitst）。

4.2　门户中 Web Application 集成

本节将介绍创建 iView 的步骤，该 iView 显示后台系统中运行的 Web Dynpro ABAP 应用

程序。

1）按照路径（http://xitst:50000/irj）登录门户系统，如图 4-1 所示。



图 4-1

在当前文件夹下，用鼠标右键单击“Context”菜单中的 NSP，按照图 4-2 所示的路径启动 iView 向导。



图 4-2

2）在应用技术（Application Technology）向导页中，选择 Web Dynpro ABAP 应用程序，创建一个 Web Dynpro ABAP 应用程序的 iView，如图 4-3 所示。



图 4-3

注：此处可选择“Web Dynpro 应用程序”单选按钮来创建 iView，本例截图中均使用“视图模板”选项中的页面配置。

单击 下一个 按钮，如图 4-4 所示。



图 4-4

选中“SAP Web Dynpro iView”单选按钮，单击下一个按钮，如图 4-5 所示。



图 4-5

填写必要信息，单击下一个按钮，如图 4-6 所示。

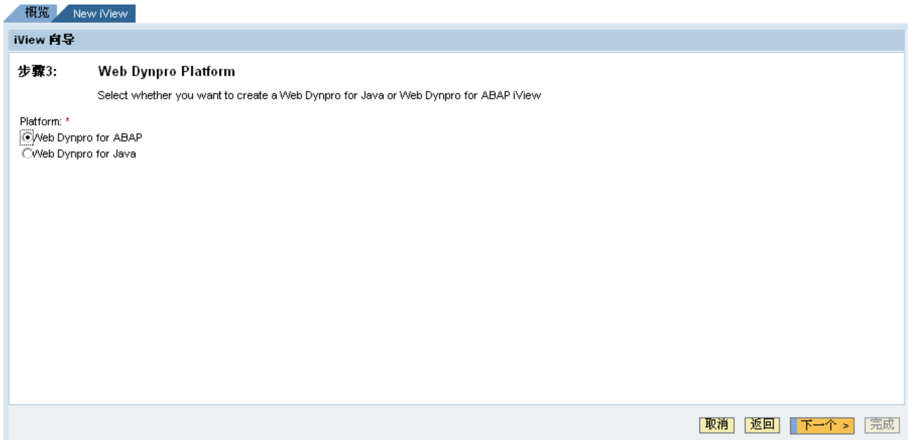


图 4-6

选中“Web Dynpro for ABAP”单击按钮，单击下一个按钮，如图 4-7 所示。

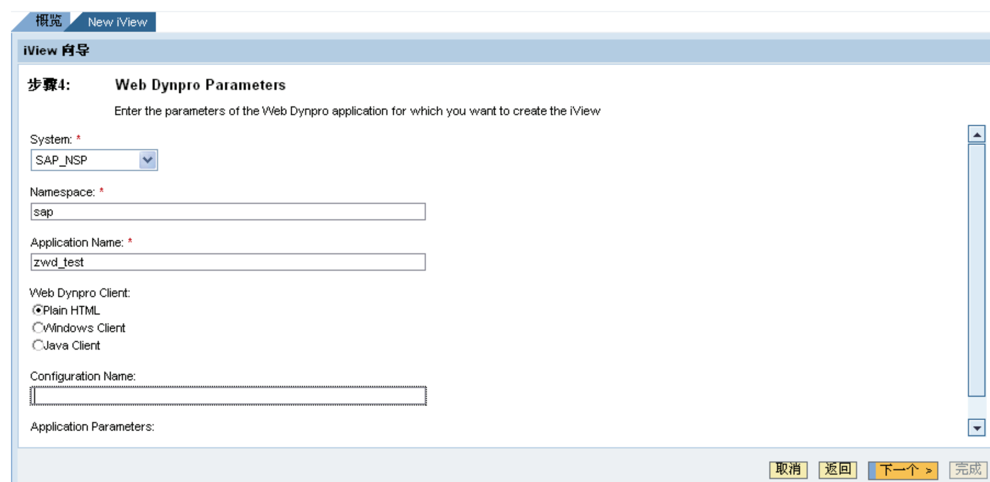


图 4-7

填写必要信息，单击下一个按钮，如图 4-8 所示。



图 4-8

确认信息无误，单击完成按钮，生成的结果如图 4-9 所示。

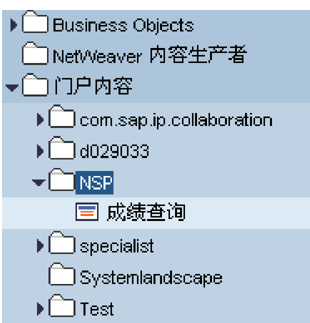


图 4-9

3）选中 iView 成绩查询，用鼠标右键单击 Context 菜单中的预览，iView 显示结果如图 4-10 所示。

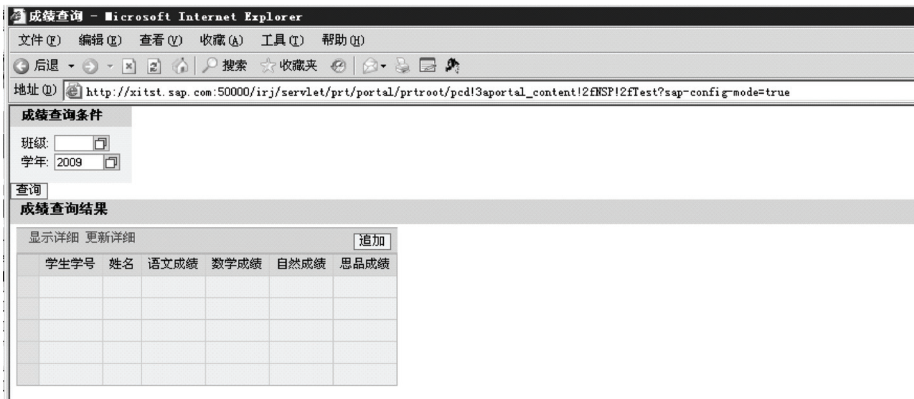


图 4-10

4.3 门户事件（Portal Events）

在 SAP Enterprise Portal 中，程序员可以在相同的门户页面上处理不同应用类型的特殊 iView。在这里，iView 可以是使用不同技术的应用（如 Web Dynpro ABAP/Java 或 BSP）。这些 iView 之间的通信可以通过事件功能门户事件（或客户端事件）实现。

一个 Web Dynpro ABAP 应用程序可注册门户网站事件。以这种方式，Web Dynpro 应用程序可以处理在门户网站的另一个 iView 触发的事件。因此，这里的应用程序基于什么样的技术与 iView 无关。因为事件处理程序在被调用时分配，这个事件发生时，该分配被存储在 Web Dynpro 应用程序中，该应用程序已注册其本身到门户网站的事件中。

与注册事件一样，Web Dynpro 应用程序可以触发任何门户网站的事件。在这种情况下，该事件由各自的 iView 传递到门户网站。门户网站把事件传送到已注册此事件的所有 iView。反过来，基于 iView 的程序最终处理的事件可以用其他技术触发，当然也可以是 Web Dynpro 应用程序。

注：在 iView 之间的门户事件功能都发生在同一浏览器窗口中。在不同的浏览器窗口之间的 iView 不能接收对应事件。

所有参与事件传递的 iView 也必须属于同一个域的。否则，由于 JavaScript 的限制，门户网站事件不能工作。

4.3.1 触发门户事件

在一个 Web Dynpro 应用程序所在 iView 内触发门户事件，然后该事件被传递到门户网站的一个或多个其他的 iView，并且该门户网站将事件传给已注册此事件的所有 iView。以这种方式，事件可以在不同技术开发的 iView 之间传输数据。

实现门户事件之间的 iView 必须在同一浏览器窗口中。不同的浏览器窗口的 iView 中事

件之间不能被触发。

所有参加触发或相应的事件也必须属于同一个域的。否则，门户网站事件不能工作，由于 JavaScript 的限制。

在 Web Dynpro ABAP 中，门户管理器（接口 IF_WD_PORTAL_INTEGRATION）提供了触发方法，程序员可以根据应用程序的要求用 Web Dynpro 代码向导将这个方法调用的模板插入到源代码中。

```
METHOD ONACTIONFIRE_PORTAL_EVENT .
    DATA: L_API_COMPONENT TYPE REF TO IF_WD_COMPONENT,
    L_PORTAL_MANAGER TYPE REF TO IF_WD_PORTAL_INTEGRATION.
    L_API_COMPONENT = WD_COMP_CONTROLLER->WD_GET_API( ).
    L_PORTAL_MANAGER = L_API_COMPONENT->GET_PORTAL_MANAGER( ).
    L_PORTAL_MANAGER->FIRE(
    PORTAL_EVENT_NAMESPACE = 'my_namespace_for_Web_Dynpro_documentation'
    PORTAL_EVENT_NAME = 'showCustomer'
    PORTAL_EVENT_PARAMETER = CUSTOM_ID ).
ENDMETHOD.            "ONACTIONFIRE_PORTAL_EVENT
```

除了强制性参数 Namespace 和 Name，也可以传递其余定制参数，见表 4-1。

表 4-1

说　　明	接收字符串
事件的命名空间	PORTAL_EVENT_NAMESPACE
事件名称	PORTAL_EVENT_NAME
参数	PORTAL_EVENT_PARAMETER

程序员可以从 Web Dynpro 应用程序中的任何位置触发这样的门户事件。该事件被发送到客户端，甚至还可以在一个请求响应周期内触发几个门户网站的事件。

通常情况下是在 Web Dynpro 应用程序的一个动作处理程序来触发一个门户事件。例如，用一个动作处理程序的 UI 元素（如一个按钮）触发门户事件。当一个门户网站的事件被触发时，首先通过一个内部的应用程序将事件从 iView 传递到门户，同时会被门户内的一个或几个 iView 处理。

事件的命名空间和名称的字符限制于 SAP 企业门户、客户端架构的命名空间。只可以使用表 4-2 中列出的字符。

表 4-2

大写字母	"A" - "Z"
小写字母	"a" - "z"
数值	"0" - "9"
特殊字符	"-" "_"　"."

注：

- 必须首先用字符串 urn 命名空间。

- 命名空间 com.sapportals.portal 和 com.sapportals 作为 SAP 保留命名空间，因此不能使用它们作为自己应用程序的命名空间。
- 命名空间和名称是区分大小写的，如 com.sap.webdynpro.testApplications.testEvent。

4.3.2 注册并处理事件

（1）注册门户事件

要注册 Web Dynpro 应用程序的门户网站事件，可用接口 IF_WD_PORTAL_INTEGRATION 的方法 SUBSCRIE_EVENT。

注：

- 门户网站管理器中删除已注册的门户网站事件，使用方法 UNSUBSCRIBE_EVENT。
- 门户事件的注册登记或删除操作必须在每个视图各自方法 WDDOINIT 中进行。

使用的 Web Dynpro 代码向导生成一个模板。程序员可以根据实际编辑注册代码。

```
METHOD WDDOINIT .
    DATA: L_API_COMPONENT TYPE REF TO IF_WD_COMPONENT,
    L_PORTAL_MANAGER TYPE REF TO IF_WD_PORTAL_INTEGRATION,
    VIEW TYPE REF TO IF_WD_VIEW_CONTROLLER.
    L_API_COMPONENT = WD_COMP_CONTROLLER->WD_GET_API( ).
    L_PORTAL_MANAGER = L_API_COMPONENT->GET_PORTAL_MANAGER( ).
    VIEW ?= WD_THIS->WD_GET_API( ).
    L_PORTAL_MANAGER->SUBSCRIBE_EVENT(
    PORTAL_EVENT_NAMESPACE =
 '   my_namespace_for_Web_Dynpro_documentation'
    PORTAL_EVENT_NAME = 'showCustomer'
    VIEW = VIEW
    ACTION = 'RECIEVE_CUSTOMER_ID' ).
ENDMETHOD.                "WDDOINIT
```

输入事件的命名空间和名称。命名空间和事件名称相结合的字符串必须是唯一的。另外，输入的动作ACTION，该动作用来触发门户网站的事件。

注：动作 RECEIVE_CUSTOMER_ID 在视图中的“Actions”选项卡中创建，用它可以判断门户事件的触发点。

（2）处理门户事件

一个门户事件参数传递的方法是通过操作参数 WDEVENT 的函数 GET_STRING 进行的。通过可选参数 PORTAL_EVENT_PARAMETER，可以取得应用程序传递给处理程序方法的信息。在下面的示例中，一个特定客户的 ID，其值被传递到组件控制器的方法 SHOWCUSTOMER。

```
METHOD ONACTIONRECIEVE_CUSTOMER_ID .
    DATA: EVT_NAME TYPE STRING,
    CUST_ID TYPE SCUSTOM-ID.
    EVT_NAME = WDEVENT->GET_STRING( NAME = 'PORTAL_EVENT_NAME' ).
```

```
            IF EVT_NAME = 'showCustomer'.
              CUST_ID = WDEVENT->GET_STRING( NAME = 'PORTAL_EVENT_PARAMETER' ).
              WD_COMP_CONTROLLER->SHOWCUSTOMER( CUSTOMER_ID = CUST_ID ).
            ENDIF.
        ENDMETHOD.              "ONACTIONRECIEVE_CUSTOMER_ID
```

4.4 事件应用实例

本节的示例将通过一个页面显示班级条件查询和班级群体查询结果（两个 iView 分别为不同的应用程序中的视图），在两个视图之间通过事件参数传递值。

4.4.1 创建应用程序

（1）查询应用程序

1）启动 ABAP 工作台中的 Web Dynpro 编辑界面，创建 Web Dynpro 组件。

选中“Web Dynpro Comp./Intf.”，在相应的文本框中输入程序的名称，如图 4-11 所示。

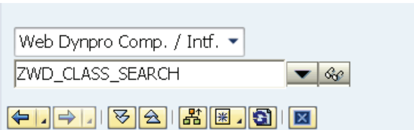


图 4-11

创建 Web Dynpro 组件如图 4-12 所示

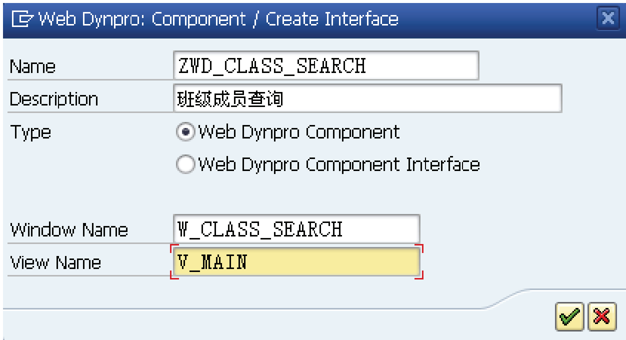


图 4-12

编辑 Web Dynpro 组件如图 4-13 所示。

Object Name	Description
ZWD_CLASS_SEARCH	班级成员查询
COMPONENTCONTROLLER	班级成员查询----组件控制器
Component Interface	
INTERFACECONTROLLER	班级成员查询----组件控制器
Interface Views	
W_CLASS_SEARCH	班级成员查询----窗体
Views	
V_MAIN	班级成员查询----视图
Windows	
W_CLASS_SEARCH	班级成员查询----窗体

图 4-13

2）在 ABAP 工作台中的视图编辑界面编辑视图。

选择视图 V_MAIN，选择“Context”选项卡，编辑 Context 如图 4-14 所示。

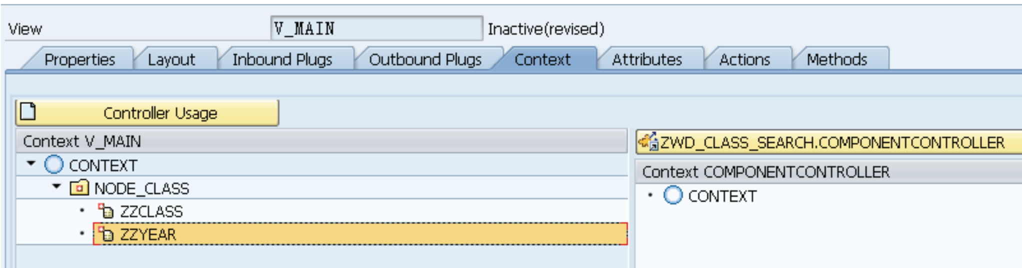


图 4-14

Context 节点 NODE_CLASS 设定如图 4-15 所示。

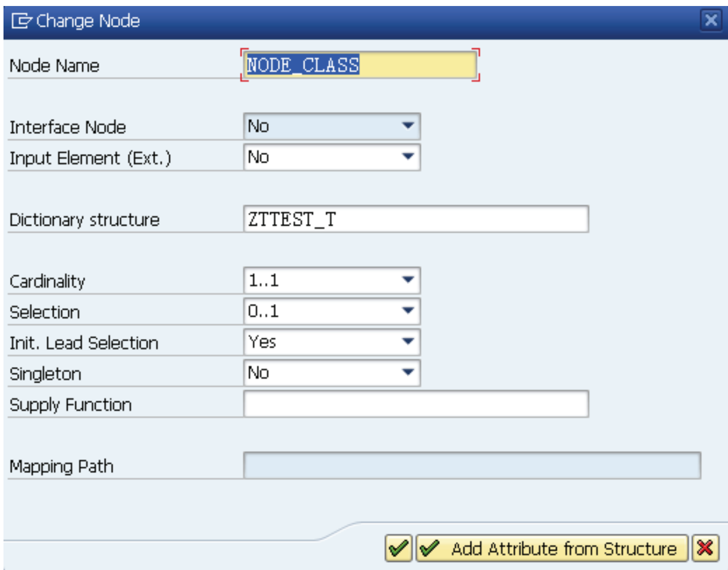


图 4-15

选中“Layout”选项卡，插入 UI 元素如图 4-16 所示。

编辑布局如图 4-17 所示。

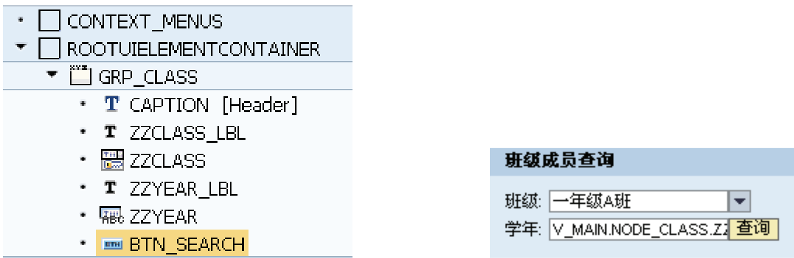


图 4-16　　　　图 4-17

为按钮添加动作如图 4-18 所示。



图 4-18

动作处理程序编辑如下：

```
METHOD ONACTIONSEARCH .
*    定义 Context 操作变量
    DATA LO_ND_NODE_CLASS TYPE REF TO IF_WD_CONTEXT_NODE.

    DATA LO_EL_NODE_CLASS TYPE REF TO IF_WD_CONTEXT_ELEMENT.
    DATA LS_NODE_CLASS TYPE WD_THIS->ELEMENT_NODE_CLASS.
    DATA LS_ZTTEST_T TYPE ZTTEST_T.
*    定义 Portal 事件操作变量
    DATA LO_API_COMPONENT  TYPE REF TO IF_WD_COMPONENT.
    DATA LO_PORTAL_MANAGER TYPE REF TO IF_WD_PORTAL_INTEGRATION.
    DATA LV_PARAMETER TYPE STRING.
*       navigate from <CONTEXT> to <NODE_CLASS> via lead selection
    LO_ND_NODE_CLASS = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->WDCTX_N
ODE_CLASS ).

*       @TODO handle non existant child
*       IF lo_nd_node_class IS INITIAL.
*       ENDIF.

*       get element via lead selection
    LO_EL_NODE_CLASS = LO_ND_NODE_CLASS->GET_ELEMENT( ).
*       @TODO handle not set lead selection
    IF LO_EL_NODE_CLASS IS INITIAL.
    ENDIF.
*       get all declared attributes
    LO_EL_NODE_CLASS->GET_STATIC_ATTRIBUTES(
        IMPORTING
           STATIC_ATTRIBUTES = LS_NODE_CLASS ).
*  取值不为空的前提下
```

```
    CHECK LS_NODE_CLASS IS NOT INITIAL.
*   判断数据是否存在
    SELECT     * FROM  ZTTEST_T
             INTO LS_ZTTEST_T UP TO 1 ROWS
             WHERE  ZZCLASS  = LS_NODE_CLASS-ZZCLASS
             AND    ZZYEAR   = LS_NODE_CLASS-ZZYEAR .

    ENDSELECT.
*   存在的场合
    IF  SY-SUBRC = 0.
    CONCATENATE LS_NODE_CLASS-ZZCLASS LS_NODE_CLASS-ZZYEAR INTO LV_PARAMETER.
      LO_API_COMPONENT = WD_COMP_CONTROLLER->WD_GET_API( ).
      LO_PORTAL_MANAGER = LO_API_COMPONENT->GET_PORTAL_MANAGER( ).

      LO_PORTAL_MANAGER->FIRE(
        PORTAL_EVENT_NAMESPACE = 'urn:com.sap.webdynpro.portalintergration.showClassNumber'
          PORTAL_EVENT_NAME      = 'showClassNumber'
          PORTAL_EVENT_PARAMETER =  LV_PARAMETER
        "PORTAL_EVENT_SCOPE     = IF_WD_PORTAL_INTEGRATION=>CO_EVENT_SCOPE-
CURRENT_WINDOW
          ).
    ENDIF.
ENDMETHOD.
```

3）在 ABAP 工作台中的 Web Dynpro Application 编辑界面，创建 Web Dynpro Application，如图 4–19 所示。

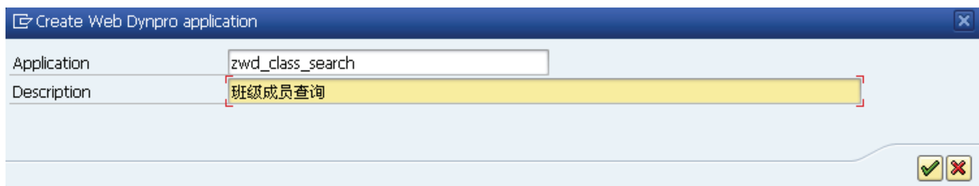


图 4–19

（2）查询结果应用程序

1）启动 ABAP 工作台，在 Web Dynpro 编辑界面创建 Web Dynpro 组件。

选择“Web Dynpro Comp./Intf.”，在相应的文本框中输入程序的名称，如图 4–20 所示。

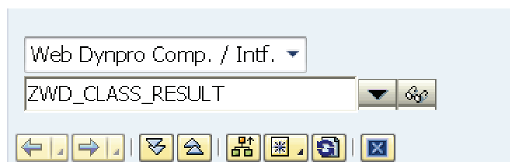


图 4–20

创建 Web Dynpro 组件如图 4–21 所示。

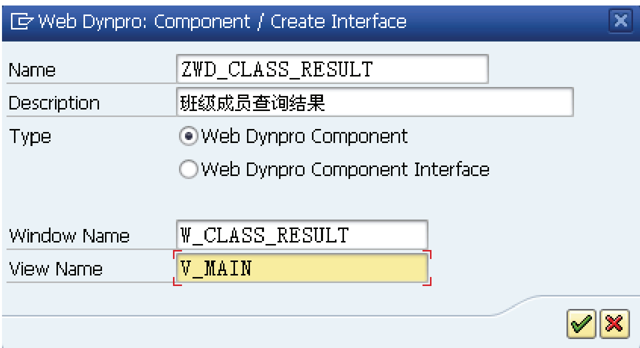


图 4–21

编辑 Web Dynpro 组件如图 4–22 所示。

Object Name	Description
ZWD_CLASS_RESULT	班级成员查询结果
COMPONENTCONTROLLER	班级成员查询结果----组件控制器
Component Interface	
INTERFACECONTROLLER	班级成员查询结果----组件控制器
Interface Views	
W_CLASS_RESULT	班级成员查询结果----窗体
Views	
V_MAIN	班级成员查询结果----视图
Windows	
W_CLASS_RESULT	班级成员查询结果----窗体

图 4–22

2）在 ABAP 工作台中的视图编辑界面编辑视图。

选择视图 V_MAIN，选择“Context”选项卡，编辑 Context 如图 4–23 所示。

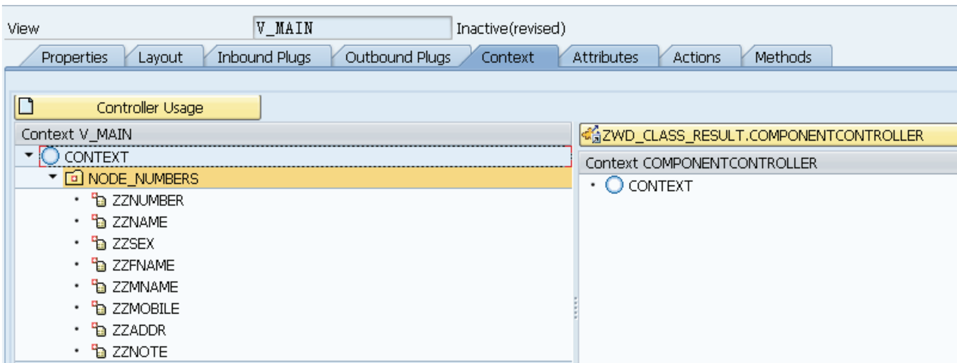


图 4–23

Context 节点 NODE_CLASS 设定如图 4–24 所示

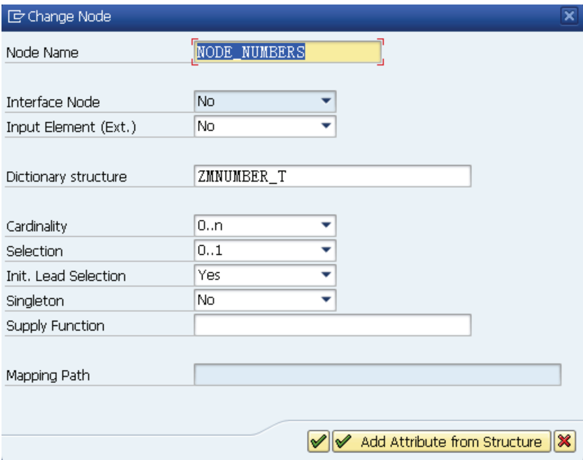


图 4-24

选择“Layout”选项卡，插入 UI 元素如图 4-25 所示。

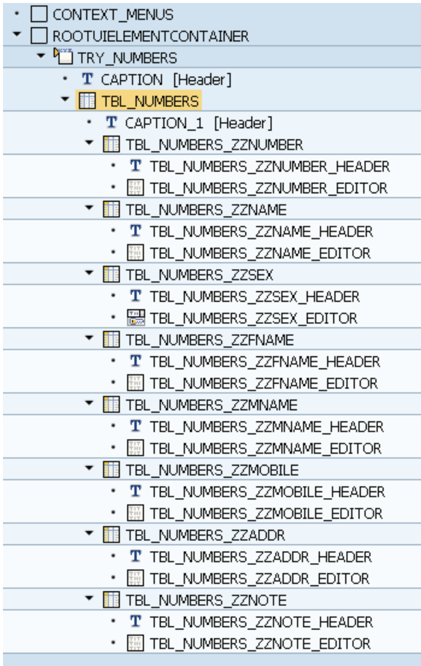


图 4-25

编辑布局如图 4-26 所示

班级成员列表

学生番号	姓名	性别	父亲	母亲	联系电话	家庭住址
V_MAIN.NODE_NUMBERS.ZZNUMBER	V_MAIN.NODE_NUMBERS.ZZNAME	...	V_MAIN.NODE_NUMBERS.ZZFNAME	V_MAIN.NODE_NUMBERS.ZZMNAME	V_MAIN.NODE_NUMBERS.ZZMOBILE	V_MAIN.NODE_NUMBERS.ZZADDR
V_MAIN.NODE_NUMBERS.ZZNUMBER	V_MAIN.NODE_NUMBERS.ZZNAME	...	V_MAIN.NODE_NUMBERS.ZZFNAME	V_MAIN.NODE_NUMBERS.ZZMNAME	V_MAIN.NODE_NUMBERS.ZZMOBILE	V_MAIN.NODE_NUMBERS.ZZADDR
V_MAIN.NODE_NUMBERS.ZZNUMBER	V_MAIN.NODE_NUMBERS.ZZNAME	...	V_MAIN.NODE_NUMBERS.ZZFNAME	V_MAIN.NODE_NUMBERS.ZZMNAME	V_MAIN.NODE_NUMBERS.ZZMOBILE	V_MAIN.NODE_NUMBERS.ZZADDR
V_MAIN.NODE_NUMBERS.ZZNUMBER	V_MAIN.NODE_NUMBERS.ZZNAME	...	V_MAIN.NODE_NUMBERS.ZZFNAME	V_MAIN.NODE_NUMBERS.ZZMNAME	V_MAIN.NODE_NUMBERS.ZZMOBILE	V_MAIN.NODE_NUMBERS.ZZADDR
V_MAIN.NODE_NUMBERS.ZZNUMBER	V_MAIN.NODE_NUMBERS.ZZNAME	...	V_MAIN.NODE_NUMBERS.ZZFNAME	V_MAIN.NODE_NUMBERS.ZZMNAME	V_MAIN.NODE_NUMBERS.ZZMOBILE	V_MAIN.NODE_NUMBERS.ZZADDR

行 1 的 5

图 4-26

UI 元素 TBL_NUMBERS 绑定属性设定如图 4–27 所示。

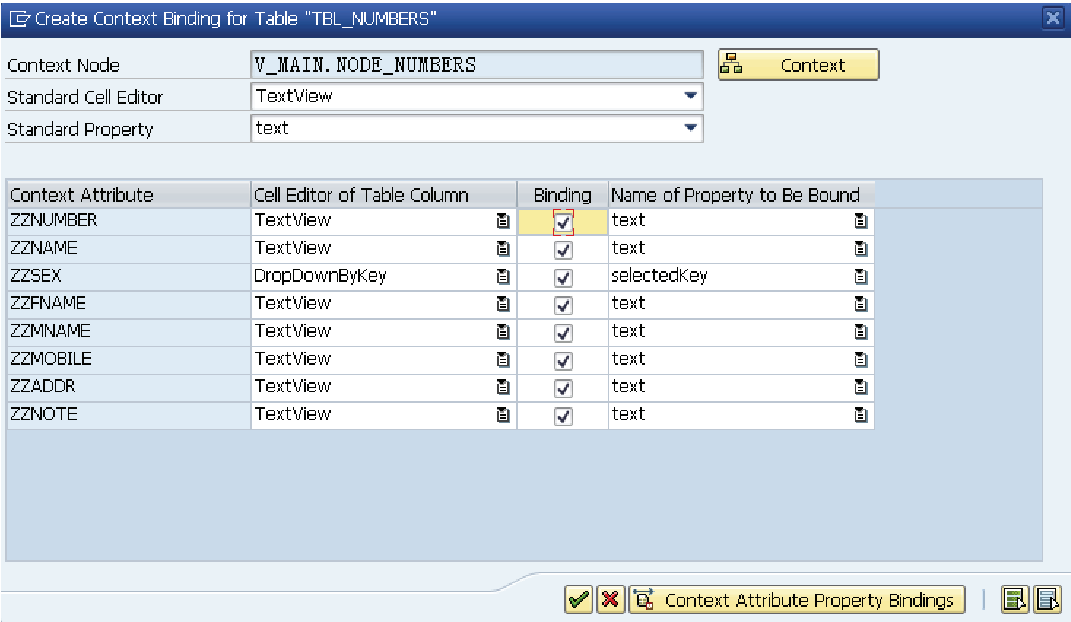


图 4–27

选中“Actions”选项卡，创建动作如图 4–28 所示。

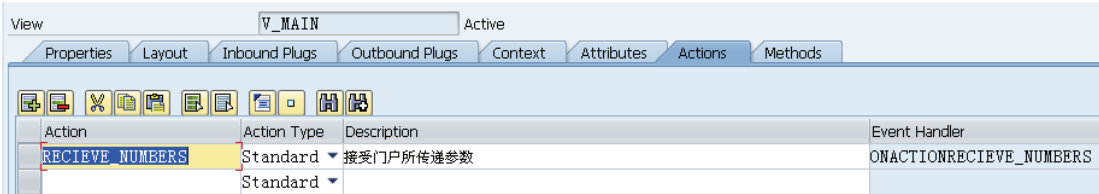


图 4–28

编辑动作处理程序如下：

```
METHOD ONACTIONRECIEVE_NUMBERS .
*   Portal 事件操作变量
    DATA: EVT_NAME TYPE STRING,
              LV_PARAMETER TYPE STRING.
*   Context 赋值操作变量
    DATA LO_ND_NODE_NUMBERS TYPE REF TO IF_WD_CONTEXT_NODE.
    DATA LT_NODE_NUMBERS TYPE WD_THIS->ELEMENTS_NODE_NUMBERS.
*   取得 Portal 事件
    EVT_NAME = WDEVENT->GET_STRING( NAME = 'PORTAL_EVENT_NAME' ).
 IF EVT_NAME = 'showClassNumber'.
  LV_PARAMETER=WDEVENT->GET_STRING(NAME='PORTAL_EVENT_PARAMETER' ).
```

```
*   navigate from <CONTEXT> to <NODE_NUMBERS> via lead selection
    LO_ND_NODE_NUMBERS = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->
WDCTX_NODE_NUMBERS ).
*   @TODO handle non existant child
*   IF lo_nd_node_numbers IS INITIAL.
*   ENDIF.

*   @TODO compute values
    SELECT   B~ZZNUMBER
          B~ZZNAME
          B~ZZSEX
          B~ZZFNAME
          B~ZZMNAME
          B~ZZMOBILE
          B~ZZADDR
          B~ZZNOTE
          FROM  ZTTEST_T AS  A
          INNER JOIN ZMNUMBER_T AS B
          ON      A~ZZSTDNO = B~ZZNUMBER
          INTO CORRESPONDING FIELDS OF TABLE LT_NODE_NUMBERS
          WHERE  ZZCLASS  =  LV_PARAMETER+0(2)
          AND    ZZYEAR   =  LV_PARAMETER+2(4)
          .
*   赋值
    LO_ND_NODE_NUMBERS->BIND_TABLE( NEW_ITEMS = LT_NODE_NUMBERS SET_IN
ITIAL_ELEMENTS = ABAP_TRUE ).

  ENDIF.
ENDMETHOD.
```

转到 Methods 标签页，编辑钩子方法 WDDOINIT 如下：

```
METHOD WDDOINIT .
* Portal 事件注册变量
  DATA: L_API_COMPONENT TYPE REF TO IF_WD_COMPONENT,
      L_PORTAL_MANAGER TYPE REF TO IF_WD_PORTAL_INTEGRATION,
      VIEW TYPE REF TO IF_WD_VIEW_CONTROLLER.
  L_API_COMPONENT = WD_COMP_CONTROLLER->WD_GET_API( ).
  L_PORTAL_MANAGER = L_API_COMPONENT->GET_PORTAL_MANAGER( ).
  VIEW ?= WD_THIS->WD_GET_API( ).
* 注册事件
  L_PORTAL_MANAGER->SUBSCRIBE_EVENT(
  PORTAL_EVENT_NAMESPACE = 'urn:com.sap.webdynpro.portalintergration.showClassNumber'
  PORTAL_EVENT_NAME = 'showClassNumber'
  VIEW = VIEW
  ACTION = 'RECIEVE_NUMBERS' ).
ENDMETHOD.
```

3）在 ABAP 工作台中的 Web Dynpro Application 编辑界面创建 Web Dynpro Application，如图 4-29 所示。

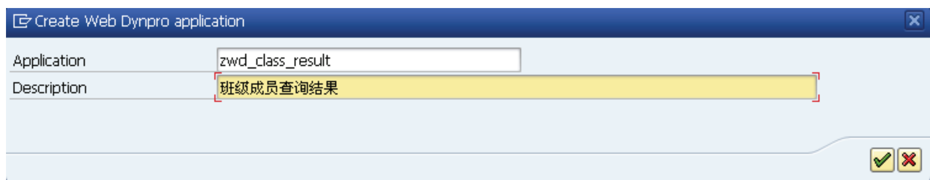


图 4-29

4.4.2 创建 iView 及页面

（1）为查询应用程序创建 iView

1）按照路径（http://xitst:50000/irj）登录门户系统。

转到内容管理页面，如图 4-30 所示。



图 4-30

在当前文件夹下，用鼠标右键单击“Context”菜单中的“NSP”，按照图 4-31 所示的路径启动 iView 向导。

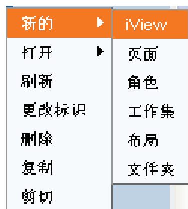


图 4-31

2）在应用技术（Application Technology）向导页中，选择 Web Dynpro ABAP 应用程序，创建一个 Web Dynpro ABAP 应用程序的 iView，如图 4-32 所示。



图 4-32

单击下一个按钮，如图 4-33 所示。



图 4-33

选中“SAP Web Dynpro iView”单选按钮，单击下一个按钮，如图 4-34 所示。



图 4-34

填写必要信息，单击下一个按钮，如图 4-35 所示。

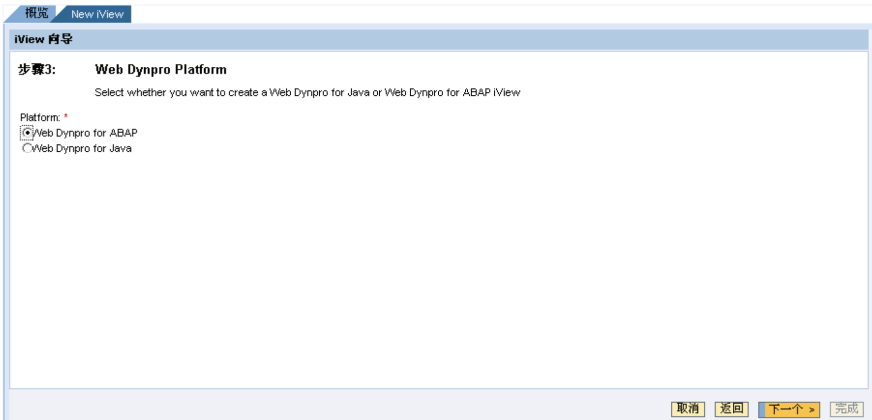


图 4-35

选中“Web Dynpro for ABAP”单选按钮，单击下一个按钮，如图 4-36 所示。

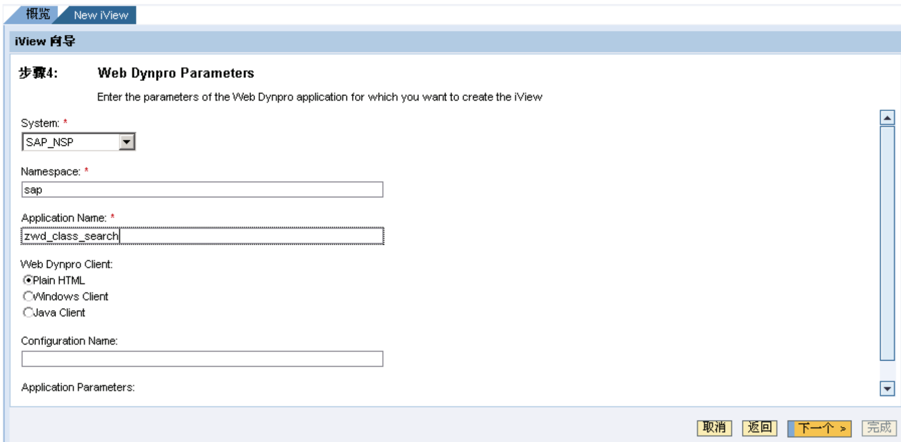


图 4-36

填写必要信息，单击下一个按钮，如图 4-37 所示。



图 4-37

确认信息无误，单击完成按钮。

（2）为查询结果应用程序创建 iView

1）在当前文件夹下，用鼠标右键单击“Context”菜单中的“NSP”，按照图 4-38 所示的路径启动 iView 向导。



图 4-38

2）在应用技术（Application Technology）向导页中，选择 Web Dynpro ABAP 应用程序，创建一个 Web Dynpro ABAP 应用程序的 iView，如图 4-39 所示。



图 4-39

单击下一个按钮，如图 4-40 所示。



图 4-40

选中“SAP Web Dynpro iView”单选按钮，单击下一个按钮，如图 4-41 所示。



图 4-41

填写必要信息，单击 下一个 按钮，如图 4-42 所示。



图 4-42

选中“Web Dynpro for ABAP”单选按钮，单击 下一个 按钮，如图 4-43 所示。

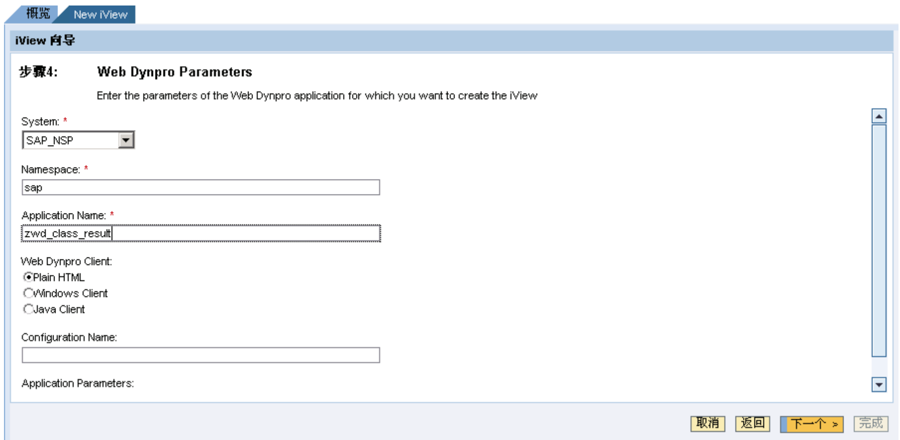


图 4-43

填写必要信息，单击 下一个 按钮，如图 4-44 所示。



图 4-44

确认信息无误，单击 完成 按钮。

（3）为创建 iView 创建页面

1）在当前文件夹下，用鼠标右键单击“Context”菜单中的“NSP”，按照图 4-45 所示的路径启动页面向导。



图 4-45

2）在应用技术（Application Technology）向导页中，填写必要信息，创建框架为两列的页面，页面属性设置如图 4-46 所示。



图 4-46

单击 下一个 按钮，如图 4-47 所示。

选中“Web Dynpro 代理页”单选按钮，单击 下一个 按钮，如图 4-48 所示。



图 4–47

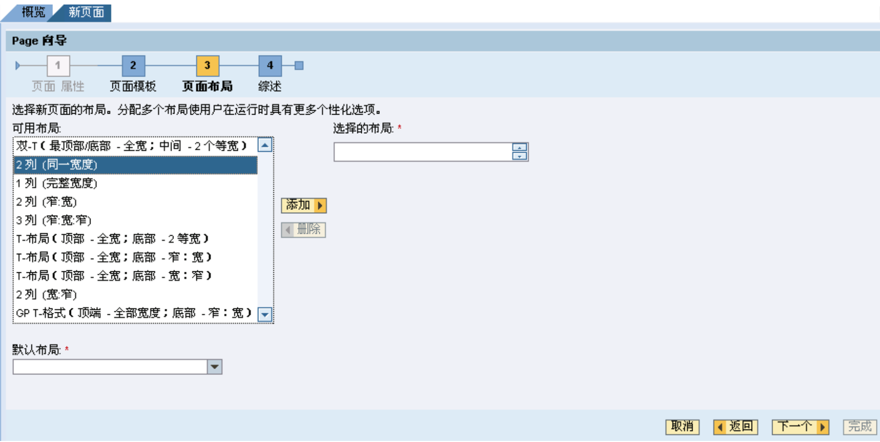


图 4–48

选择布局 2 列（同一宽度），单击添加按钮，为页面选择 iView 的布局，如图 4–49 所示。



图 4–49

单击 下一个 按钮，如图 4-50 所示。

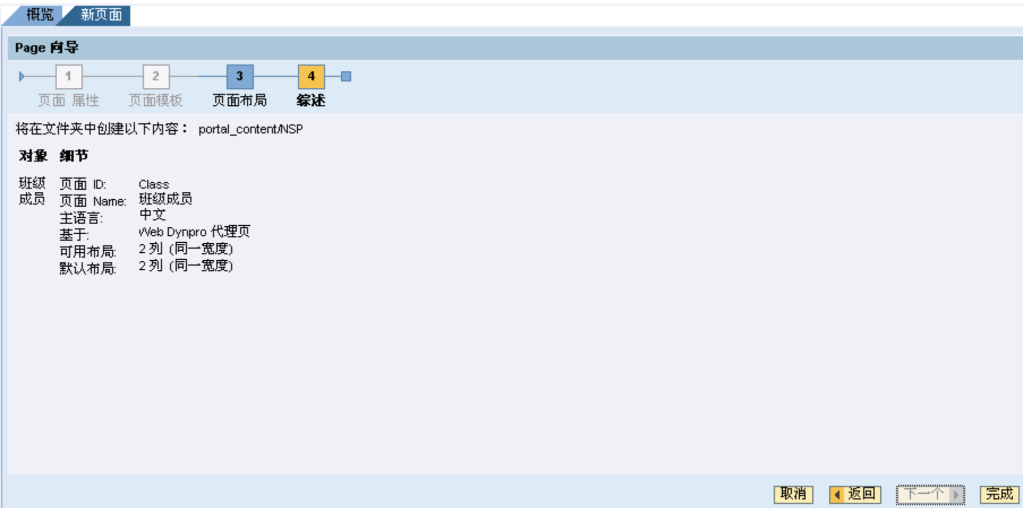


图 4-50

确认信息无误，单击 完成 按钮。

3）编辑页面属性。

页面编辑完成后，单击 确定 按钮，编辑页面 Page，如图 4-51 所示。



图 4-51

在当前文件夹下，选中 iView 班级成员查询画面 和 班级成员查询结果画面，用鼠标右键单击“Context”菜单，按照图 4-52 所示的路径将 iView 添加到页面。

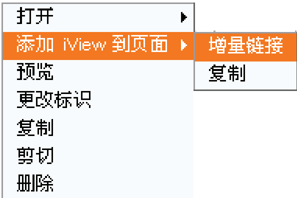


图 4-52

页面属性设定如图 4-53 所示。



图 4-53

选择各个 iView，单击属性按钮，选择各列对应的 iView 内容，将其高度类型设为 AUTOMATIC，并单击“保存”按钮，如图 4-54 所示。

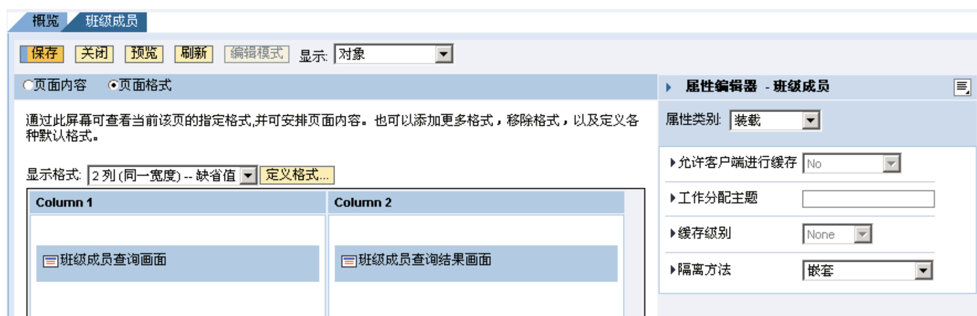


图 4-54

编辑结果如图 4-55 所示。



图 4-55

4）测试页面上事件响应。

测试结果如图 4-56 所示。



图 4-56

4.5 门户导航（Portal Navigation）

SAP 企业门户网站支持各类门户内容之间的导航。例如，一个 Web Dynpro 应用程序可以浏览门户网站的内容以及其他不同的 Web Dynpro 应用程序。门户网站的内容可以是一个 BSP 或 ITS 中的应用。

页面导航是基于对象的，可以是绝对的或相对的。

4.5.1 基于对象的导航（OBN）

SAP 企业门户内容的结构是基于角色的。用户可以采用顶级导航和详细的导航浏览特定用户的导航结构。门户网站的导航可以把不同的 iView 或页面（Page）作为一个门户网站的内容并在它们之间进行导航。

在许多情况下，需要充分使用相对导航或绝对导航，导航到一个特定的 iView 或页面。特别地，有时需要更多灵活性。为了这个目的，程序员可以使用基于对象的导航。它允许程序员在一个更高的层次（业务对象）定义导航步骤。不是定义一个具体的 URL 目标，而是调用一个特定业务对象的特定操作。

在门户中可以配置具体的 iView（或页面）用于执行此操作。这种配置可以是特定的角色或特定的用户。Web Dynpro 应用程序本身传递业务对象的名称和链接到它的操作。

在 Web Dynpro ABAP 中，基于对象的导航集成是非常类似门户网站事件的集成。为了触发导航本身，Web Dynpro Framework 提供了一个服务。该服务可以从应用程序调用，像门户网站事件，是门户管理的一部分。

1．触发基于对象的导航

可以在 Web Dynpro ABAP 的门户网站中调用接口 IF_WD_PORTAL_INTEGRATION 的方法 NAVIGATE_TO_OBJECT，激活基于对象的导航。程序员可以使用 Web Dynpro 中的代码向导自动生成以下代码：

```
DATA LR_COMPONENTCONTROLLER TYPE REF TO IG_COMPONENTCONTROLLER .
DATA L_API_COMPONENTCONTROLLER TYPE REF TO IF_WD_COMPONENT.
DATA LR_PORT_MANAGER TYPE REF TO IF_WD_PORTAL_INTEGRATION.
LR_COMPONENTCONTROLLER = WD_THIS->GET_COMPONENTCONTROLLER_CTR( ).
L_API_COMPONENTCONTROLLER = LR_COMPONENTCONTROLLER->WD_GET_API( ).
LR_PORT_MANAGER=L_API_COMPONENTCONTROLLER->GET_PORTAL_MANAGER( ).
CALL METHOD LR_PORT_MANAGER->NAVIGATE_TO_OBJECT
   EXPORTING
    SYSTEM             = NAVIGATION_DATA-SYSTEM
    OBJECT_TYPE        = NAVIGATION_DATA-OBJECT
    OPERATION          = NAVIGATION_DATA-OPERATION
    OBJECT_VALUE_NAME  = NAVIGATION_DATA-OBJECT_VALUE_NAME
    OBJECT_VALUE       = NAVIGATION_DATA-OBJECT_VALUE
    BUSINESS_PARAMETERS = BUS_PARAMETER_LIST
    FORWARD_OBN_METADATA = NAVIGATION_DATA-FORWARD_OBN_METADATA.
```

（1）对于导航只有以下两个参数是必需的

1）SYSTEM：指定业务对象被分配到的系统（或系统别名）。

2）OBJECT_TYPE：指定使用的业务对象。

（2）所有其他参数都是可选的

1）OBJECT_VALUE。通常有许多不同的业务对象实例，如对于业务对象客户。可以使用参数指定特定客户（如使用客户编号）调用基于对象的导航步骤。

2）OPERATION：可以使用该参数指定基于对象的导航步骤的操作。

3）OBEJCT_VALUE_NAME：指定对象的值传送一个 URL 参数到 OBN 步骤。该参数的标准名称是 ObjectValue。程序员也可以更改该名称。

4）BUSINESS_PARAMETERS：除了指定对象的值，可以定义由 OBN 步骤转发的其他参数。可以定义一个参数字符串，如 Mode=Edit&ShowHeader=false。在业务对象的操作做好相应准备的前提下，这些参数可以被目标 OBN 使用。

5）FORWARD_OBN_METADATA。有时，从当前的导航步骤中接收更多的详细信息对于 OBN 目标是有益的。举例来说，如果程序员实现一个应用程序，用于实施某一业务对象上执行的不同操作，则应用程序必须知道所引发 OBN 步骤的操作。因此，可以通过以下参数：

- obn.system：业务对象所分配到的系统。
- obn.bo_type：业务对象本身
- obn.operation：当前操作，该值是“_default_”。

（3）在门户中维护目标应用

目标应用程序保存在门户网站中的业务对象的相应的操作中，这通常由门户管理员维护。

对于一个 WDA 目标应用程序的业务参数，使用 Web Dynpro ABAP 应用程序定义的参数时，如果动态地分配应用程序参数给启动插头，则可以在应用程序中的文档中找到这些参数。

为了使 Web Dynpro ABAP 应用到目标应用程序之间能够正确传输业务参数，下面的 JavaScript 代码必须被保存在目标应用程序的基于对象导航内：

```
return 'DynamicParameter=' + objValue;
```

2．基于角色的权限检查

因此，导航的执行依赖于在门户中自定义角色的设置。例如，一个角色的用户可以具有用于显示和编辑页面内容的授权，而用户可能只被允许的另一个作用是要显示门户内容。如果用户要触发基于对象的导航，但该角色不具有相应操作的授权，则相应的错误消息会被显示出来。为了尽可能地提供友好的用户界面，从一开始就不提供会出现此问题的用户操作。然而，为了这个目的，必须由门户得到相应操作的授权信息。这可以通过 Web Dynpro ABAP 系统中的类 CL_WDR_PORTAL_OBNWEB_SERVICE 调用 Web 服务来取得。

4.5.2　绝对地址导航

可以在 Web Dynpro ABAP 系统所在的门户中使用门户管理器（接口 IF_WD_PORTAL_INTEGRATION 的方法 NAVIGATE_ABSOLUTE）激活绝对地址导航。程序员可以使用代码向导生成一个模板以便编辑具体应用。

使用绝对导航工具，必须知道要显示的页面名称，以便把该名称传到方法中。

```
DATA LR_COMPONENTCONTROLLER TYPE REF TO IG_COMPONENTCONTROLLER .
DATA L_API_COMPONENTCONTROLLER TYPE REF TO IF_WD_COMPONENT.
DATA LR_PORT_MANAGER TYPE REF TO IF_WD_PORTAL_INTEGRATION.
LR_COMPONENTCONTROLLER = WD_THIS->GET_COMPONENTCONTROLLER_CTR( ).
L_API_COMPONENTCONTROLLER = LR_COMPONENTCONTROLLER->WD_GET_API( ).
LR_PORT_MANAGER=L_API_COMPONENTCONTROLLER->GET_PORTAL_MANAGER( ).
CALL METHOD LR_PORT_MANAGER->NAVIGATE_ABSOLUTE
    EXPORTING
        NAVIGATION_TARGET   = NAVIGATION_DATA-TARGET
        NAVIGATION_MODE     = NAVIGATION_DATA-NAVIGATION_MODE
        WINDOW_FEATURES     = NAVIGATION_DATA-WINDOW_FEATURES
        WINDOW_NAME         = NAVIGATION_DATA-WINDOW_NAME
        HISTORY_MODE        = NAVIGATION_DATA-HISTORY_MODE
        TARGET_TITLE        = NAVIGATION_DATA-TARGET_TITLE
        CONTEXT_URL         = NAVIGATION_DATA-CONTEXT_URL
        POST_PARAMETERS     = ABAP_FALSE
        USE_SAP_LAUNCHER    = ABAP_TRUE
        BUSINESS_PARAMETERS = BUS_PARAMETER_LIST
        LAUNCHER_PARAMETERS = LAUNCHER_PARAMETER_LIST.
```

导航目标是这里的唯一必选参数。它代表了在门户网站中的绝对地址。其他参数被用于控制导航，是可选的。可以在门户中的各个应用程序启动器中设置业务参数和参数。为了业务参数能正确传递到目标应用程序，必须设置参数 USE_SAP_LAUNCHER。如果它是一个 SAP 应用程序（如 BSP 或 Web Dynpro 等），则必须将其值设置为 TRUE，见表 4-3。

表 4-3

参　数	可选	值		描述
NAVIGATION_TARGET		地址 如 ROLES://portal_content/ web_dynpro_abap/ web_dynpro_abap_tester/ portal_integration/ portalNavigation/ portal_navigation_target		绝对地址、页面或 iView 在门户内容目录的路径。该路径将显示在门户网站目录。例如，当编辑一个页面或 iView 的路径
NAVIGATION_MODE	√	INPLACE	在同一页上显示导航目标	导航模式
		EXTERNAL	在新的页面显示导航，但只是作为目标的 iView	
		EXTERNAL_PORTAL	在新的门户页面显示导航目标	
WINDOW_FEATURES	√	TOOLBAR	显示标准工具栏	附加的 JavaScript 窗口参数。例如，字符集或尺寸规格，如宽度=300 高度=250，这些参数是由逗号分隔。不允许有空格
		LOCATION	显示的网站地址	
		DIRECTORIES	显示浏览器中 Directory 按钮	
		STATUS	显示状态栏	
		MENUBAR	显示菜单栏	
		SCROLLBARS	显示滚动条	
		RESIZABLE	Windows 可以调整大小	
		WIDTH	窗体的宽度	
		HEIGTH	窗体的高度	
WINDOW_NAME	√	字符串		外部浏览器目标页面的标题
HISTORY_MODE	√	ALLOW_DUPLICATIONS	可以列出一次以上的导航历史	指定导航历史记录中是否列出访问过的网页地址
		NO_DUPLICATIONS	只能列出一次导航历史	
		NO_HISTORY	没有历史导航条目	
TARGET_TITLE	√	字符串		门户页面的标题
CONTEXT_URL	√	字符串		指定所使用的导航 Context
POST_PARAMETERS	√	TRUE	作为一个 POST 请求传送参数	参数传输选项
		FALSE（默认值）	作为一个 GET 请求的传输参数	
USE_SAP_LAUNCHER	√	TRUE（默认值）	目标调用使用SAP 发射器，如BSP	SAP 发射器
		FALSE	目标调用不使用 SAP 发射器	
BUSINESS_PARAMETERS	√	WDR_NAME_VALUE_LIST		用 URL 传递导航到目标应用程序的参数
LAUNCHER_PARAMETERS	√	WDR_NAME_VALUE_LIST		应用发射器的参数列表，如 WebDynpro Namespace=myNamespace

注：如果在 Web Dynpro 应用程序中定义 BUSINESS_PARAMETERS 为应用程序的参数，并且参数的名称以“APP”作为字头，则它们将被自动转发到 Web Dynpro 应用程序的启动插头被标记为启动参数。在这种情况下，作为导航目标的 iView/页面必须被分配给用户相应角色。如果没有分配，则导航可能不被触发。

4.5.3 相对地址导航

相对地址导航变体可以用于提供跨几个目录内容显示。在门户网站中 Web Dynpro ABAP 应用要激活相对地址导航，使用接口 IF_WD_PORTAL_INTEGRATION 的方法 NAVIGATE_RELATIVE。使用代码向导可以生成一个模板，程序员可以根据具体应用改变相关参数。

```
CALL METHOD LR_PORT_MANAGER->NAVIGATE_RELATIVE
EXPORTING
BASE_URL = NAVIGATION_DATA-BASE_URL
LEVELS_UP = LEVELS_UP
PATH = PATHLIST
NAVIGATION_MODE = NAVIGATION_DATA-NAVIGATION_MODE
WINDOW_FEATURES = NAVIGATION_DATA-WINDOW_FEATURES
WINDOW_NAME = NAVIGATION_DATA-WINDOW_NAME
HISTORY_MODE = NAVIGATION_DATA-HISTORY_MODE
TARGET_TITLE = NAVIGATION_DATA-TARGET_TITLE
CONTEXT_URL = NAVIGATION_DATA-CONTEXT_URL
USE_SAP_LAUNCHER = ABAP_TRUE
BUSINESS_PARAMETERS = BUS_PARAMETER_LIST
LAUNCHER_PARAMETERS = LAUNCHER_PARAMETER_LIST.
```

相对地址导航可以使用自定义的起点，具体参数见表 4-4。

表 4-4

参　数	可　选	值	描　述
BASE_URL	√	路径名	相对导航的起点
LEVELS_UP		数值	导航结构中向上的导航步骤
PATH		路径名	目标应用程序的相对路径列表

示例：

```
BASE_URL : pcd:role1/folder1/folder2/fodler3/workset1/page1
LEVELS_UP: 3
PATHLIST: folder4/workset2/page2
目标路径 pcd:role1/folder1/folder2/folder4/workset2/page2
```

参数 BASE_URL 可以不指定，在这种情况下，当前的 URL 被使用。PATH 是一个最近的节点列表。在这种情况下，PATH 列表中的值将是：

```
folder4
workset2
page2
```

4.5.4 Resume Plugs 和门户导航

对于每一个接口视图，程序员可以标记一个入站插头为继续插头（Resume plug）。在门户导航时，继续插头（Resume plug）的作用是：在运行的应用程序如果在门户导航时识别这样的插头，前者不是导航时退出，而是被转移到挂起模式。会话（Session）ID 和 Context 内容被保留下来，并开始一个新的应用程序。如果一个用户通过门户网站的导航菜单，然后返回到原来的应用程序，则它可以保留它离开时完全相同的状态。

如果应用程序使用了继续插头，则应用程序导航时，应用程序相应会话（Session）会保留在门户内。然而，在许多情况下，完全关闭导航的应用程序可能是明智的，也是绝对必要的。在这种情况下，不能为接口视图配备继续插头（Resume plug）。

（1）方法 WDDOAPPLICATIONSTATECHANGE

方法 WDDOAPPLICATIONSTATECHANGE 为组件控制器的方法。应用程序的状态发生变化时，该方法被处理，即当应用程序由运行状态转换为暂停状态（由挂起插头中断应用）时，或者在随后通过继续插头重新启动。该方法从而取代了 WDDOEXIT 方法。

应用程序开发可以（并不是必需的）添加代码到方法 WDDOAPPLICATIONSTATECHANGE 中。它可能是有用的，如在程序挂起时删除数据库的锁。

当应用程序运行在门户网站中，如果在设计时没有实现挂起操作，门户导航需要谨慎使用挂起的操作。继续插头（Resume plug）的使用只能确保某个应用程序的中断，而不会导致门户导航退出。但是，如果应用程序没有运行在门户中，上述情况并不会发生，此时可以不利用门户网站的导航机制。

（2）门户之外的 Resume Plugs 和 Suspend Plugs

如果用户想在一个纯粹的 Web Dynpro 导航中，中断应用程序并保留会话，则必须为接口视图相关的窗口创建一个挂起插头（Suspend Plugs），并在适当的时候触发它。每个接口视图可以有多个标记为挂起插头的出站插头。触发挂起插头也导致方法 WDDOAPPLICATIONSTATECHANGE 的调用，方法 WDDOEXIT 则不会被调用。挂起的应用程序的 URL 会自动传递给随后的应用程序，所以 Web Dynpro 导航中可以通过随后的应用程序返回到原来的应用程序。

如果在 Web Dynpro ABAP 应用程序中出站插头被标记为挂起标记，则这可能会导致在门户中应用程序的运行时错误。如果不能完全确定应用程序的执行是在门户网站之外，则应该避免使用挂起插头。

4.6 导航应用实例

4.6.1 业务对象创建

1）按照路径（http://xitst:50000/irj）登录门户系统，如图 4-57 所示。



图 4-57

在当前文件夹下，用鼠标右键单击“Context”菜单中的“Business Objects”，按照图 4-58 所示的路径新建文件夹。

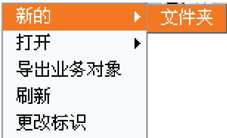


图 4-58

启动向导如图 4-59 所示。



图 4-59

填写相关信息，单击完成按钮。

2）从“Context”菜单中选中新建文件夹“私有业务对象”，按照图 4-60 所示的路径创建业务对象。

业务对象添加页面如图 4–61 所示。



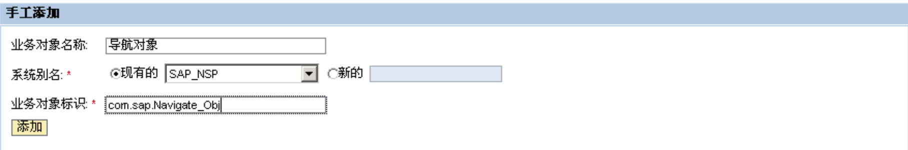


图 4–60　　　　　　　　　　　　　　　　图 4–61

填写名称、标识等信息，单击添加按钮，结果如图 4–62 所示。



图 4–62

单击保存按钮，保存新建业务对象。

3）编辑新建业务对象，为其添加工序，如图 4–63 所示。



图 4–63

单击添加按钮，如图 4–64 所示。



图 4-64

填写名称、标识等信息，单击保存按钮，编辑结果如图 4-65 所示。

4）在门户内容文件夹内，选择 iView成绩查询，为新添加工序分配 iView。

选择 iView，单击鼠标右键，按照以下路径打开编辑页面，如图 4-66 所示。

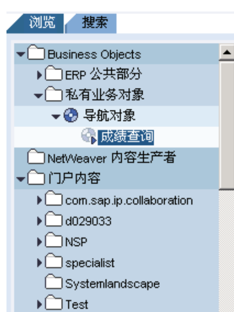


图 4-65

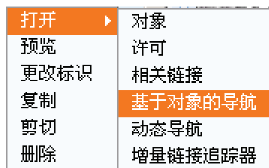


图 4-66

基于对象的导航页面显示如图 4-67 所示。

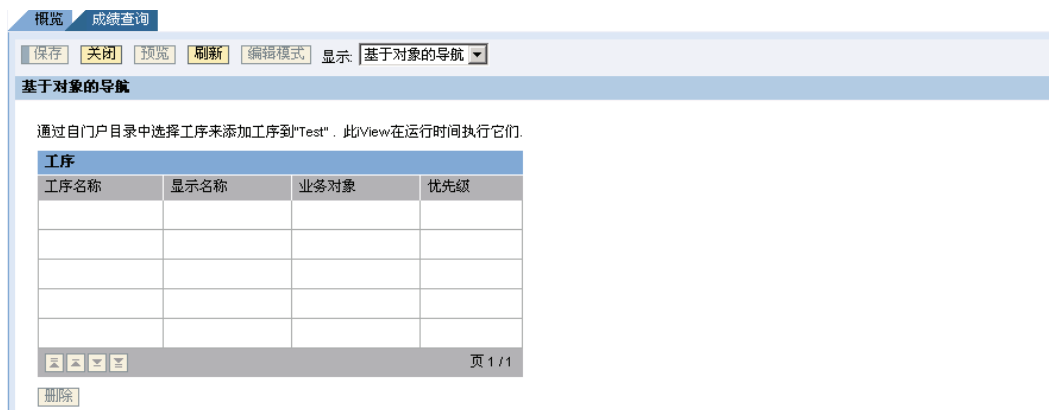


图 4-67

从“Context”菜单中选中新建工序成绩查询，单击鼠标右键，按照图 4-68 所示的路径将工序添加到 iView。



图 4–68

结果如图 4–69 所示。

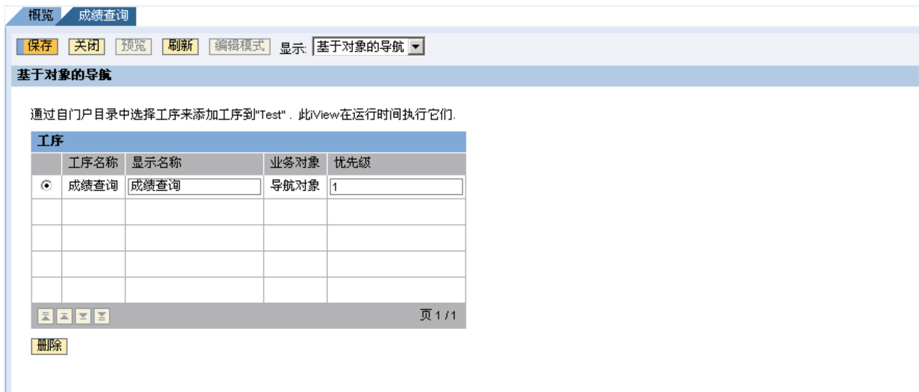


图 4–69

单击保存按钮。

注：该过程也可以将业务对象打开，选中要操作的工序，单击iView，根据相关菜单将iView分配给工序。

4.6.2　应用程序创建

1）ABAP 工作台 编辑画面 创建 Web Dynpro 组件

选中“Web Dynpro Comp./Intf.”，在相应的文本框中输入程序的名称，如图 4–70 所示。

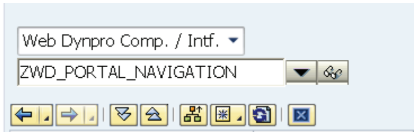


图 4–70

创建 Web Dynpro 组件如图 4–71 所示。

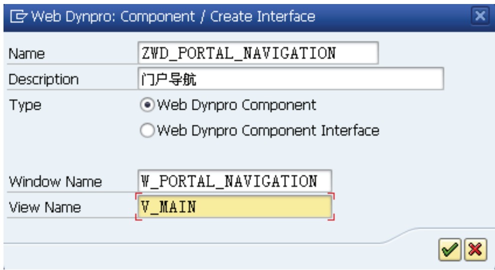


图 4–71

编辑 Web Dynpro 组件如图 4-72 所示。

Object Name	Description
ZWD_PORTAL_NAVIGATION	门户导航
COMPONENTCONTROLLER	门户导航----组件控制器
Component Interface	
INTERFACECONTROLLER	门户导航----组件控制器
Interface Views	
W_PORTAL_NAVIGA	门户导航----窗体
Views	
V_MAIN	门户导航----视图
Windows	
W_PORTAL_NAVIGATIC	门户导航----窗体

图 4-72

2）在 ABAP 工作台中的编辑界面编辑视图。

选择视图 V_MAIN，选中“Layout”选项卡，插入 UI 元素如图 4-73 所示。

CONTEXT_MENUS
ROOTUIELEMENTCONTAINER
LTA_OBJECT
LTA_ABSOLUTE
LTA_RELATIVE

图 4-73

编辑布局如图 4-74 所示。

对象导航
绝对导航
相对导航

图 4-74

为 LinkToAction 类 UI 元素 LTA_OBJECT 创建动作如图 4-75 所示。

Create Action

Component ZWD_PORTAL_NAVIGATION
View V_MAIN
Action TO_OBJECT
Description 对象导航
Transfer UI Event Parameters

图 4-75

动作处理程序编辑如下：

```
METHOD ONACTIONTO_OBJECT .
 DATA LO_API_COMPONENT  TYPE REF TO IF_WD_COMPONENT.
 DATA LO_PORTAL_MANAGER TYPE REF TO IF_WD_PORTAL_INTEGRATION.
     LO_API_COMPONENT = WD_COMP_CONTROLLER->WD_GET_API( ).
     LO_PORTAL_MANAGER = LO_API_COMPONENT->GET_PORTAL_MANAGER( ).
     LO_PORTAL_MANAGER->NAVIGATE_TO_OBJECT(
          SYSTEM                 = 'SAP_NSP'
          OBJECT_TYPE            = 'com.sap.Navigate_Obj'
*          OBJECT_VALUE_NAME       = 'objectValue'
*          OBJECT_VALUE            = OBJECT_VALUE
         OPERATION              = 'Test'
*          BUSINESS_PARAMETERS      = BUSINESS_PARAMETERS
*          FORWARD_OBN_METADATA      = ABAP_FALSE
*          USE_FORWARDPARAMETERS_OF_AL = ABAP_FALSE
*          RESOLVING_MODE          = CO_RESOLVING_MODE-SOURCE_ROLE
            ).
ENDMETHOD.
```

为 LinkToAction 类 UI 元素 LTA_ABSOLUTE 创建动作如图 4-76 所示。

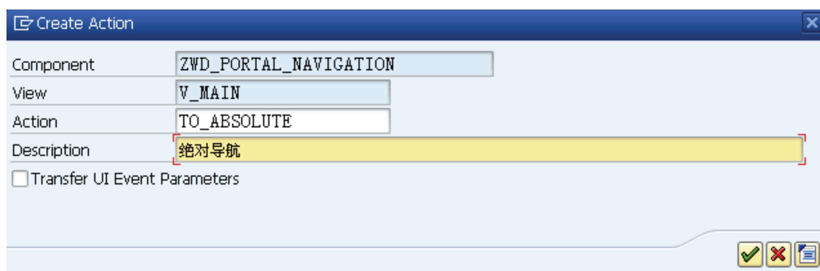


图 4-76

动作处理程序编辑如下：

```
METHOD ONACTIONTO_ABSOLUTE .
    DATA LO_API_COMPONENT  TYPE REF TO IF_WD_COMPONENT.
    DATA LO_PORTAL_MANAGER TYPE REF TO IF_WD_PORTAL_INTEGRATION.

    LO_API_COMPONENT = WD_COMP_CONTROLLER->WD_GET_API( ).
    LO_PORTAL_MANAGER = LO_API_COMPONENT->GET_PORTAL_MANAGER( ).
    LO_PORTAL_MANAGER->NAVIGATE_ABSOLUTE(
         NAVIGATION_TARGET   = 'ROLES://portal_content/NSP/Test'
         NAVIGATION_MODE   = IF_WD_PORTAL_INTEGRATION=>CO_SHOW_INPLACE
*         WINDOW_FEATURES     = WINDOW_FEATURES
*         WINDOW_NAME       = WINDOW_NAME
*         HISTORY_MODE      = IF_WD_PORTAL_INTEGRATION=>CO_NO_DUPLICATES
*         TARGET_TITLE      = TARGET_TITLE
*         CONTEXT_URL       = CONTEXT_URL
*         POST_PARAMETERS     = ABAP_FALSE
```

```
*         USE_SAP_LAUNCHER    = ABAP_TRUE
*         BUSINESS_PARAMETERS = BUSINESS_PARAMETERS
*         LAUNCHER_PARAMETERS = LAUNCHER_PARAMETERS
            ).
ENDMETHOD.
```

为 LinkToAction 类 UI 元素 LTA_RELATIVE 创建动作如图 4-77 所示。

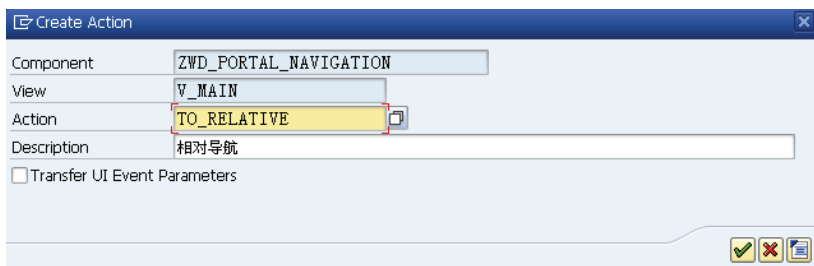


图 4-77

动作处理程序编辑如下：

```
METHOD ONACTIONTO_RELATIVE .
  DATA LO_API_COMPONENT  TYPE REF TO IF_WD_COMPONENT.
  DATA LO_PORTAL_MANAGER TYPE REF TO IF_WD_PORTAL_INTEGRATION.

  LO_API_COMPONENT = WD_COMP_CONTROLLER->WD_GET_API( ).
  LO_PORTAL_MANAGER = LO_API_COMPONENT->GET_PORTAL_MANAGER( ).
  DATA:LT_PATH  TYPE STRING_TABLE , LV_PATH  TYPE STRING.
    LV_PATH  = 'NSP'.
  APPEND LV_PATH  TO LT_PATH .
    LV_PATH  =  'Test'.
  APPEND LV_PATH  TO LT_PATH .

  LO_PORTAL_MANAGER->NAVIGATE_RELATIVE(
      BASE_URL           = 'ROLES://portal_content/NSP/Navigation'
      LEVELS_UP          = '2'
      PATH               = LT_PATH
*         NAVIGATION_MODE   = IF_WD_PORTAL_INTEGRATION=>CO_SHOW_INPLACE
*         WINDOW_FEATURES     = WINDOW_FEATURES
*         WINDOW_NAME         = WINDOW_NAME
*         HISTORY_MODE        = IF_WD_PORTAL_INTEGRATION=>CO_NO_DUPLICATES
*         TARGET_TITLE        = TARGET_TITLE
*         CONTEXT_URL         = CONTEXT_URL
*         USE_SAP_LAUNCHER    = ABAP_TRUE
*         BUSINESS_PARAMETERS = BUSINESS_PARAMETERS
*         LAUNCHER_PARAMETERS = LAUNCHER_PARAMETERS
*         POST_PARAMETERS     = ABAP_FALSE
            ).
```

```
    ENDMETHOD.
```

3）在 ABAP 工作台中的编辑界面创建 Web Dynpro Application 如图 4-78 所示。

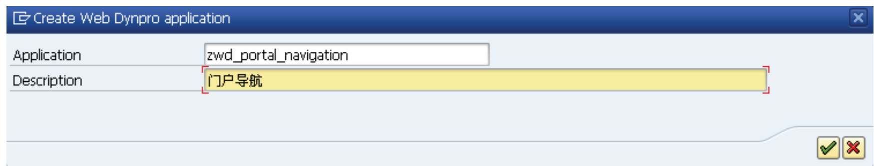


图 4-78

创建 Web Dynpro 应用程序的 iView：

在应用技术（Application Technology）向导页中，选择 Web Dynpro ABAP 应用程序，创建一个 Web Dynpro ABAP 应用程序的 iView，如图 4-79 所示。



图 4-79

单击下一个按钮，如图 4-80 所示。



图 4-80

选中“SAP Web Dynpro iView”单选按钮，单击下一个按钮，如图 4-81 所示。



图 4-81

填写必要的信息，单击下一个按钮，如图 4-82 所示。



图 4-82

选中“Web Dynpro for ABAP”单选按钮，单击下一个按钮，如图 4-83 所示。

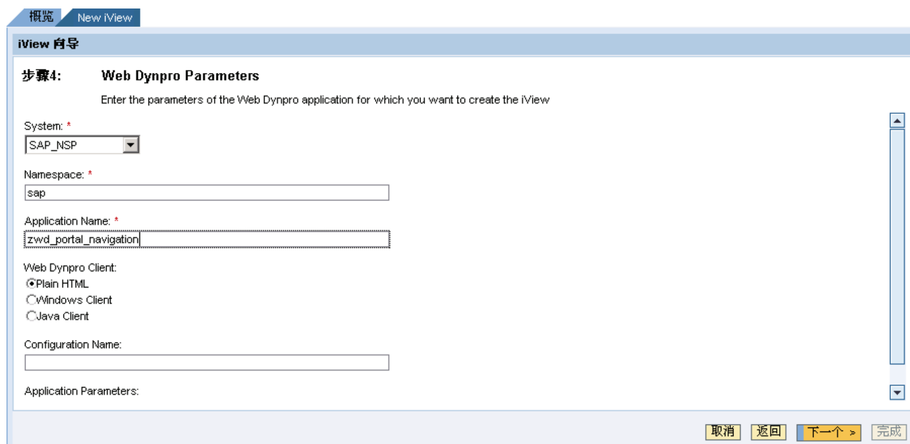


图 4-83

填写必要信息，单击[下一个]按钮，如图 4-84 所示。



图 4-84

确认信息无误，单击[完成]按钮，并将其属性“输入点”改为“是”，如图 4-85 所示。

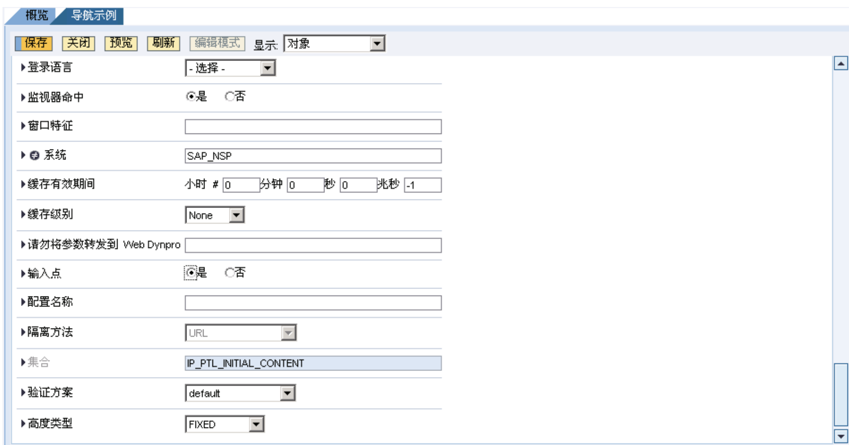


图 4-85

编辑结果如图 4-86 所示。

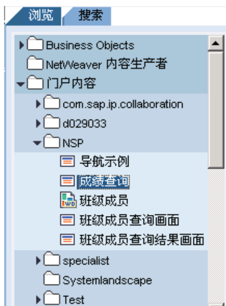


图 4-86

4.6.3 应用程序导航测试

（1）创建门户角色

1）在门户内容文件夹下，用鼠标右键单击“Context”菜单中的“NSP”，按照图 4-87 所示的路径创建角色。

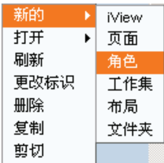


图 4-87

角色创建向导初始界面如图 4-88 所示。



图 4-88

填写必要的信息，单击下一个按钮，如图 4-89 所示。

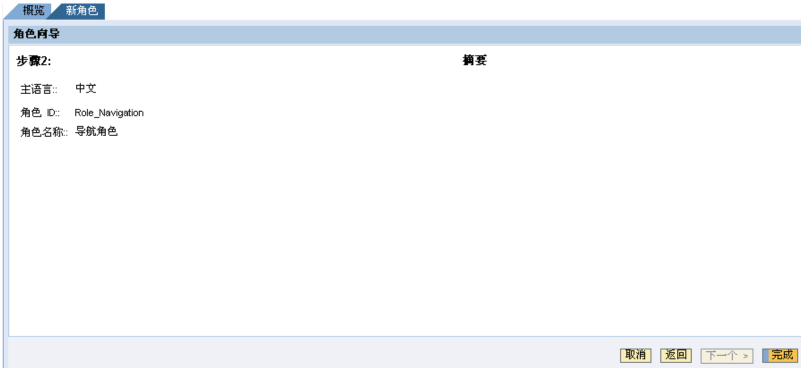


图 4-89

确认信息无误，单击完成按钮。

2）打开角色对象编辑画面，选中导航示例和成绩查询，单击鼠标右键，按照图 4-90 所示路径将 iView 添加至角色。

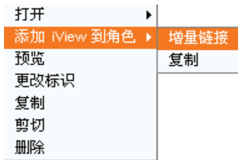


图 4–90

编辑结果如图 4–91 所示。



图 4–91

（2）分配门户角色

新建角色分配给登录用户，如图 4–92 所示。

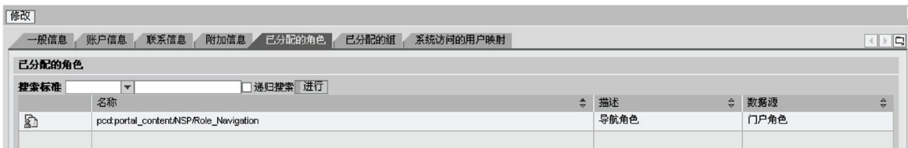


图 4–92

（3）测试门户导航

按照路径（http://xitst:50000/irj）用分配角色后的用户登录门户系统。

单击“导航示例”菜单，如图 4–93 所示。

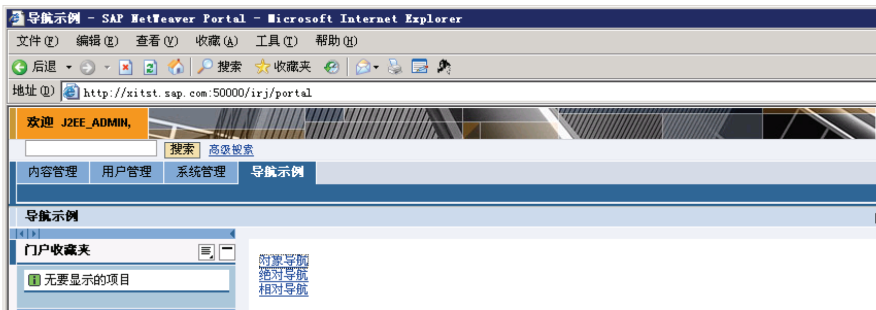


图 4–93

分别单击各个链接，导航结果如图 4-94 所示。

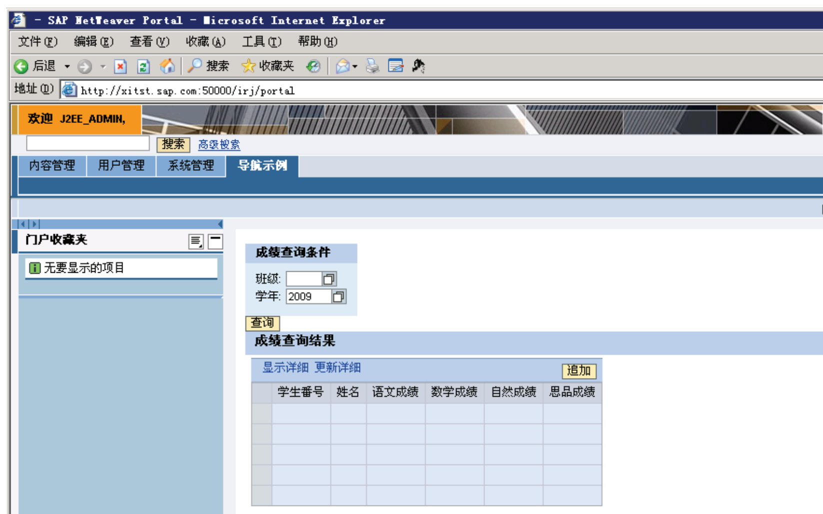


图 4-94

4.7 工作保护模式（Work Protect Mode）

工作保护模式提供了基础设施，用于处理在 SAP NetWeaver 门户中未存储的数据。如果一个应用程序所输入的数据还没有被保存，那么这个应用程序被称为“dirty”。正常情况下，数据丢失是因为当用户导航到另一个应用程序时没有事先保存数据。为了防止这种情况的发生，门户的客户端框架监控所有应用程序的当前状态。

示例：工作保护模式下的对话框，如图 4-95 所示。

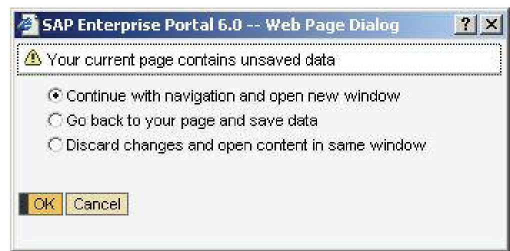


图 4-95

应用程序必须定义一个特殊的标识（dirty），当有未保存的数据时它告诉门户网站。程序员可以使用接口 IF_WD_PORTAL_INTEGRATION 中的方法 SET_APPLICATION_DIRTY_FLAG 设置和取消该状态标识（TRUE，FALSE）。如果该标识设置为 TRUE 时，则在新窗口中自动执行每个导航步骤。未保存的数据将被保留在原来的窗口。这意味着用户可以切换到原来的应用程序保存数据。

下面的源代码显示了如何设置该状态：

```
data L_COMPONENTCONTROLLER type ref to IG_COMPONENTCONTROLLER .
data L_API_COMPONENTCONTROLLER type ref to IF_WD_COMPONENT.
data L_PORTAL_MANAGER type ref to IF_WD_PORTAL_INTEGRATION.
L_COMPONENTCONTROLLER = WD_THIS->GET_COMPONENTCONTROLLER_CTR( ).
L_API_COMPONENTCONTROLLER = L_COMPONENTCONTROLLER->WD_GET_API( ).
L_PORTAL_MANAGER=L_API_COMPONENTCONTROLLER->GET_PORTAL_MANAGER( ).
call method L_PORTAL_MANAGER->SET_APPLICATION_DIRTY_FLAG
exporting
DIRTY_FLAG = TRUE | FALSE.
```

Web Dynpro 支持以下 3 种不同的工作保护模式（接口 IF_WD_PORTAL_INTEGRATION 的方法 SET_WORK_PROTECT_MODE）：

1）NONE。该值表示工作保护模式不被 Web Dynpro 应用程序所使用。在门户网站中如果导航到另一个应用程序，则未保存的数据将丢失，即使设置了“dirty”标志。

2）APPLICATION_ONLY。该值表示 Web Dynpro 应用程序本身决定是否有未保存的数据。也就是说，应用程序是否是“dirty”的。这就是为什么“dirty”的状态只在服务器端监控。即使使用这个值，也不能保证尚未传送到服务器中的数据不会丢失。

3）BOTH。该值表示客户端也检查“dirty”的状态。这确保了没有用户输入的情况下，尚未传送到服务器将会丢失。这是通过设置 SAP 企业门户中应用程序“dirty”状态，一旦用户已经输入了数据，该数据就不会丢失。

Web Dynpro 应用程序在运行期间可根据需要改变上述模式。例如，当用户从一个视图导航到另一个时可以改变模式。在一个视图上，输入字段并进行数据保存可能是有意义的。在这种情况下，该模式应定义为 BOTH 或 APPLICATION_ONLY。另一个视图上认为这种保护模式可能没有必要。在这种情况下，模式应定义为 NONE。

下面的源代码显示了如何设置该模式：

```
DATA LR_COMPONENTCONTROLLER TYPE REF TO IG_COMPONENTCONTROLLER .
DATA L_API_COMPONENTCONTROLLER TYPE REF TO IF_WD_COMPONENT.
DATA L_PORTAL_MANAGER TYPE REF TO IF_WD_PORTAL_INTEGRATION.
L_COMPONENTCONTROLLER =  WD_THIS->GET_COMPONENTCONTROLLER_CTR( ).
L_API_COMPONENTCONTROLLER = L_COMPONENTCONTROLLER->WD_GET_API( ).
L_PORTAL_MANAGER = L_API_COMPONENTCONTROLLER->GET_PORTAL_MANAGER( ).
CALL METHOD L_PORTAL_MANAGER->SET_WORK_PROTECT_MODE
  EXPORTING
    MODE  = NONE   |   APPLICATION_ONLY   |   BOTH.
```

第 5 章　集成表单（Integrating Forms）

Web Dynpro 用户界面，表单可以直接创建或使用基于 Adobe 的 Interactive Forms。为了便于用户界面的开发，可以使用 Adobe LiveCycle Designer 工具整合到开发工作台的编辑器和 Adobe UI 元素中。

对 NetWeaver 7.0（NW7.0）SPS10 以上的版本，表单格式可以集成使用 Zero Client Installation（ZCI）。

注：NetWeaver 7.0 SPS10，SAP强烈建议使用ZCI形式的XML接口。由于额外的中央功能已添加到随后的SP（输入帮助和下拉式列表框）中，SAP一般建议使用Web Dynpro ABAP最新支持包。

SAP建议始终使用表单的XML接口。现有基于ABAP字典接口形式的表单，可以用作打印形式。不建议这些表单使用互动形式。

使用基于 Adobe 的 SAP Interactive Forms 的解决方案有以下几种方式：

1）打印。该方案用于打印和显示表单。与互动场景不同，这里使用的是唯一的非输入功能的 PDF 表格（PDF 的打印表单）。

2）互动（交互）。这种情况下，表单包含输入功能的元素并可以编辑。

SAP 建议用户经过仔细检查应用要求后，判断是否使用互动表单。如果基于 PDF 的形式确实比一个正常的 Web Dynpro 视图有优势，则可以考虑使用该方法。

3）离线应用。系统中已经存在的 From 作为 MIME 文件对象的形式存储在 MIME 信息库中，或上传（文件上传）到 Web Dynpro 应用程序用于显示。

4）数字签名。创建或使用有关表格的程序很大程度上应用于前两个方案。然而，当调用应用程序时与 ACF 表单有一个本质的区别：当第一次调用表单时，一个 CAB 文件（其中包含一个在本地 Adobe Reader 的插件）会自动从服务器加载安装。即 Active Components Framework（ACF）服务，该服务允许用户在本地计算机上处理表单的内容。然而，ACF 没有所需的显示或打印的机制。

（1）客户端安装（ZCI） 和有效组件框架（ACF）

在 ISR 框架中交互表单是 ZCI 的一部分。在 ZCI 的基于 Web Dynpro ABAP 中，它们可以在运行中被用作 SAP NetWeaver SPS10。要启用该功能，表单必须包含特殊脚本，该脚本可以在表单生成器创建新的表单时插入。

到现在为止，需要在前端 PC 上安装 Active Component Framework（ACF）以便互动报表。作为 SAP NetWeaver 7.0 SPS10，ZCI 可以使用户在没有任何额外的插件情况下用 Adobe Reader 显示互动报表。ACF 因此不再需要互动表单。可以使用报表制作与 SAP NetWeaver ZCI 旧版本兼容的任何交互式表单。

（2）连接测试

测试 ZCI 运行报表 FP_CHK_REPORT。

测试 ACF 执行报表 FP_PDF_TEST_00，以确保 ADS 正确配置。如果配置是正确的，则测试程序将输出安装的 ADS 版本号。

（3）使用表单创建 Web Dynpro 应用

使用表单生成器（事务代码：SFP）可以创建和维护表单。请注意，在 Web Dynpro ABAP 中不支持兼容的智能表单接口。原则上可以在任何组件的任何视图中整合任意表单，但通常在一个单独视图中整合表单比较好。

5.1 集成表单条件

1. 一般条件

（1）安装 SAP NetWeaver AS 系统并附有 Adobe Document Services (ADS)功能

要测试连接到 ADS，运行报告 FP_PDF_TEST_00，将显示 ADS 版本信息。如果这个测试是成功的，运行报告 FP_TEST_00。选择一台打印机，选择打印预览选项。Adobe 浏览器出现，并显示一个简单的表单。

（2）语言设置

要显示表单所需的语言，用户必须指定相结合的语言和区域设置（国家），并提供必要的翻译。可以指定用户配置文件中（事务 SU52）的参数标签页的国家代码。输入 UCN 作为参数 ID，并输入所要求的国家短代码作为参数值，然后保存条目。

2. ZCI

（1）系统要求

1）SAP NetWeaver NW 7.0 SPS10 或更高版本。

2）SAPGUI Release 6.40 Support Package Level 20 或更高版本。

3）Adobe LiveCycle Designer 7.1, support package level 1 或更高版本。

（2）ZCI 表单要求

1）Adobe Reader Version 7.08 或更高版本。ZCI 和 HTTPS 需要 Adobe Reader 8.1。

2）Adobe Designer、version 7.1 SP 1 或更高版本。GUI patch level 20 必须安装。

3）交互表单属性。

- Web Dynpro ABAP 视图设计器内对于 InteractiveForm UI 元素的 enabled 属性设置为 true。
- 属性 displayType 区分 ZCI（选择 Native）和非 ZCI（选择 ACF），如图 5-1 所示。

Property	Value	Binding
Properties (InteractiveForm)		
ID	INTERACTIVE_FORM	
Layout Data	MatrixHeadData	
additionalArchives		
dataSource	V1.TEST_DATA	
displayType	native	
enabled	☑	
height	300px	
pdfSource		
readOnly	☐	
templateSource	ZKRADOBE0	
tooltip		
visible	Visible	
width	300px	
Events		
onSubmit	GO	

图 5-1

4）表单设计器（事务代码：SFP）。

- 布局类型的属性设置为 ZCI 布局，如图 5-2 所示。

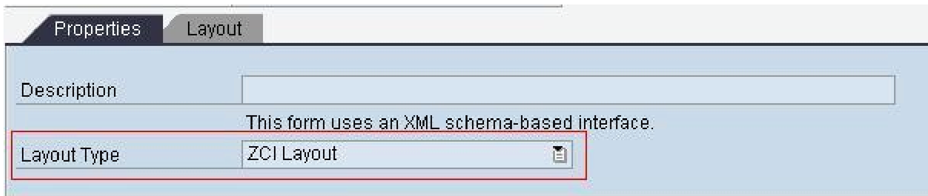


图 5-2

● 在表单中使用的 Web Dynpro Native library，如图 5-3 所示。

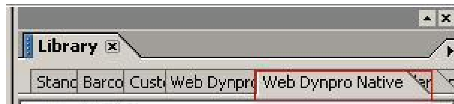


图 5-3

● 在表单中插入 ZCI 脚本实用程序，如图 5-4 所示。

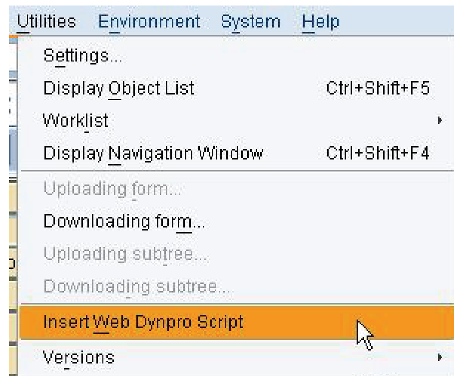


图 5-4

之后，在视图设计器中可以看到 ContainerFoundation_JS。

可能需要更新所使用的模板（使用报表程序：FP_CHK_REPORT）。

3. ACF

1）安装 SAP NetWeaver AS 系统并附有 Adobe Document Services (ADS)功能。

2）必须在本地计算机上安装 Adobe Reader（7.0 以后版本）。

5.2 在 Web Dynpro 应用中集成 PDF 表单

在程序中设置打印表单和互动表单的属性时，InteractiveForm 只有两个属性：templateSource 和 dataSource。templateSource 属性包含表单模板的信息，而绑定到 DataSource 属性的 Context 节点属性包含显示在表单上的值。

可以以 XML 格式存储或处理表单，完成在运行时由用户输入的值。要做到这一点，在视图 Context 中必须包括一个额外的属性，类型为 XSTRING，然后将其绑定到 InteractiveForm UI 元素的 pdfSource 属性。

这样就可以读取和处理 XML 格式表示的 PDF 值。使用合适的模板和 XML 数据，可以在不同的应用中创建完整的 PDF。

注：pdfSource 属性的绑定类型为 XSTRING，其内容源于后台，表单数据由后台到单独 Web Dynpro Context 节点的属性的传输不被支持。

5.3 插入互动表单

由于 Web Dynpro 应用程序中插入用于打印和互动的表单之间的差异非常小，因此这两种情况下的程序可以一起提供。

在 Web Dynpro 应用中使用静态表单，如 PDF 打印表格，表格可以用来显示或打印数据。

在互动表单的情况下，可以重复使用用户输入的内容，该内容存储在 Web Dynpro 应用程序的 Context 中。

注：在 templateSource/ dataSource 都被绑定的情况下，如果 ADS 可用，则表单才能被创建。另一方面，pdfSource 的属性在这种情况下被忽略，请注意以下几点：

1）如果将 enabled 设置为 TRUE，则显示表单通过 ACF。

2）如果 readOnly 设置为只读，则没有可以输入或修改的形式。

3）如果 enabled 选项没有设置，则没有 ACF 的形式显示，readOnly 属性将被忽略。

5.3.1 在 Web Dynpro 应用中使用 Adobe Forms

1）在 ABAP 工作台中的编辑界面创建 Web Dynpro 组件。

选中“Web Dynpro Comp./Intf.”，在相应的文本框中输入程序的名称，如图 5-5 所示。

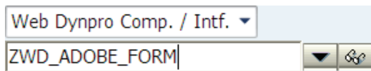


图 5-5

创建组件如图 5-6 所示。

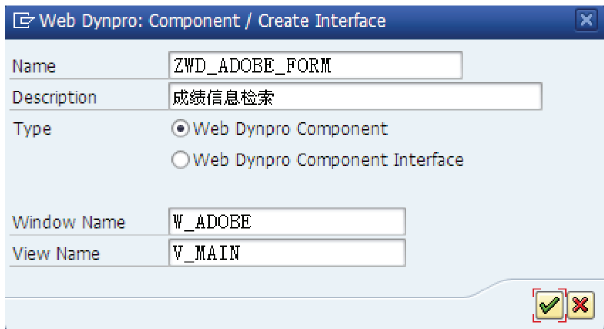


图 5-6

编辑组件如图 5-7 所示。

Object Name	Description
ZWD_ADOBE_FORM	学生信息检索
COMPONENTCONTROLLER	学生信息检索---組件控制器
Component Interface	
INTERFACECONTROLLER	学生信息检索---組件控制器
Interface Views	
W_ADOBE	学生信息检索---窗体
Views	
V_MAIN	学生信息检索---视图
Windows	
W_ADOBE	学生信息检索---窗体

图 5-7

2）在 ABAP 工作台中的编辑界面编辑 Web Dynpro 视图。

选中左侧树形菜单中的视图 V_MAIN，选择“Layout”选项卡，创建容器器类 UI 元素 GRP_TEST_KEY，添加按钮及 Interactive Form 元素，编辑布局如图 5-8 所示。

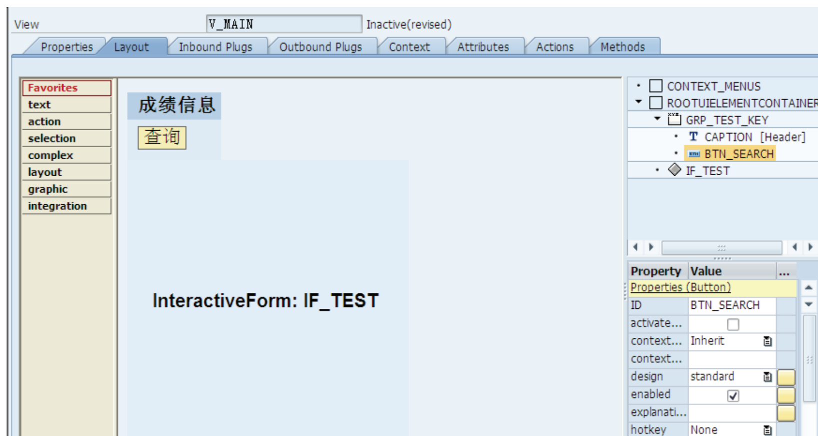


图 5-8

添加“查询”按钮动作如图 5-9 所示。



图 5-9

用 InteractiveForm 元素属性 templateSource 设定已存在 Adobe 表单 ZTEST_FORM，生成结果见表 5-1。

表 5-1

参照表单自动生成 Context 节点及属性	表单中 Context 节点及属性
Context V_MAIN CONTEXT ZTEST_FORM TEST ZZCLASS ZZYEAR ZZSTDNO ZZNAME ZZTENKOK ZZTENSAN ZZTENRIK ZZTENSHA NAME1 NAME2 YEAR CLASS	上下文 描述 ZTEST_FORM TEST 表类型 DATA ZZCLASS 班级 ZZYEAR 学年 ZZSTDNO 学生学号 ZZNAME 姓名 ZZTENKOK 语文成绩 ZZTENSAN 数学成绩 ZZTENRIK 自然成绩 ZZTENSHA 思品成绩 NAME2 姓名 NAME1 姓名 YEAR 学年 CLASS 班级

InteractiveForm 元素属性设定如图 5-10 所示。

Property	Value
Properties (InteractiveForm)	
ID	IF_TEST
addition...	
context...	Inherit
context...	
dataSou...	V_MAIN.ZTES...
displayT...	native
enabled	☑
height	300px
pdfSource	
readOnly	☐
templat...	ZTEST_FORM

图 5-10

根据生成的 Context 创建成绩查询表单如图 5-11 所示。

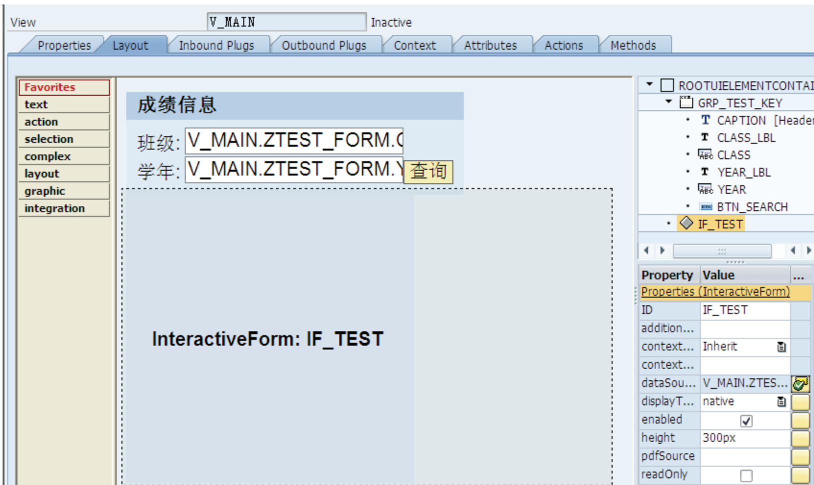


图 5-11

动作 SEARCH 的处理程序编辑如下：

```
METHOD ONACTIONSEARCH .
*   取得输入信息
   DATA LO_ND_ZTEST_FORM TYPE REF TO IF_WD_CONTEXT_NODE.
   DATA LO_EL_ZTEST_FORM TYPE REF TO IF_WD_CONTEXT_ELEMENT.
   DATA LS_ZTEST_FORM TYPE WD_THIS->ELEMENT_ZTEST_FORM.
*     navigate from <CONTEXT> to <ZTEST_FORM> via lead selection
  LO_ND_ZTEST_FORM = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->WDC
TX_ZTEST_FORM ).
*   get element via lead selection
  LO_EL_ZTEST_FORM = LO_ND_ZTEST_FORM->GET_ELEMENT( ).
*   get all declared attributes
  LO_EL_ZTEST_FORM->GET_STATIC_ATTRIBUTES(
    IMPORTING
      STATIC_ATTRIBUTES = LS_ZTEST_FORM ).
*  取得班主任姓名
  SELECT   SINGLE             A~ZTNAME
                                     B~ZTNAME
                         FROM  ZCCLASS_T  "班主任表
                       INNER JOIN ZMTNUMBER_T  AS A
                       ON  ZCCLASS_T~ZZTEANOM = A~ZTNUMBER
                       INNER JOIN ZMTNUMBER_T  AS B
                      ON ZCCLASS_T~ZZTEANOS = B~ZTNUMBER
                       INTO (LS_ZTEST_FORM-NAME1 , LS_ZTEST_FORM-NAME2)
                      WHERE  ZZCLASS  =  LS_ZTEST_FORM-CLASS .      "班级
*   get all declared attributes
  LO_EL_ZTEST_FORM->SET_STATIC_ATTRIBUTES(
    EXPORTING
         STATIC_ATTRIBUTES = LS_ZTEST_FORM ).
*取得成绩信息
   DATA LO_ND_TEST TYPE REF TO IF_WD_CONTEXT_NODE.
   DATA LT_TEST TYPE WD_THIS->ELEMENTS_TEST.
*         navigate from <CONTEXT> to <TEST> via lead selection
  LO_ND_TEST = WD_CONTEXT->PATH_GET_NODE( PATH = `ZTEST_FORM.TEST` ).
* @TODO compute values
SELECT        *   FROM  ZZTEST_V INTO CORRESPONDING FIELDS OF TABLE  LT_TEST
              WHERE  ZZCLASS   = LS_ZTEST_FORM-CLASS
             AND       ZZYEAR     = LS_ZTEST_FORM-YEAR.
*
   LO_ND_TEST->BIND_TABLE( NEW_ITEMS = LT_TEST SET_INITIAL_ELEMENTS =
ABAP_TRUE ).
ENDMETHOD.
```

3）在 ABAP 工作台中的编辑界面创建 Web Dynpro Application，如图 5-12 所示。

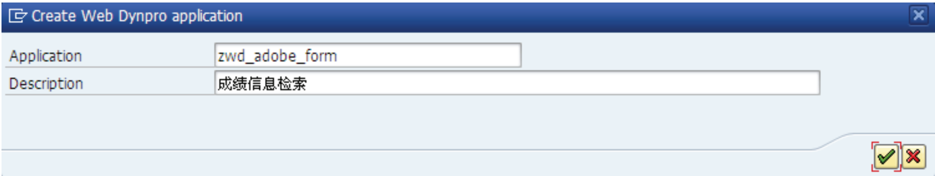


图 5-12

4）在 ABAP 工作台中的编辑界面中运行 Web Dynpro Application。测试结果如图 5-13 所示。



图 5-13

5.3.2 在 Web Dynpro 应用中新建 Adobe Forms

1）在 ABAP 工作台中的编辑界面创建 Web Dynpro 组件。

选中“Web Dynpro Comp./Intf.”，在相应的文本框中输入程序的名称，如图 5-14 所示。

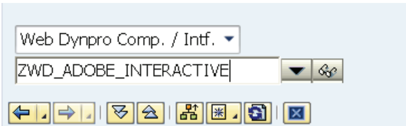


图 5-14

创建 Web Dynpro 组件如图 5-15 所示。

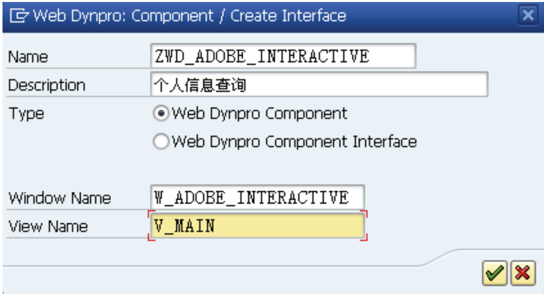


图 5-15

编辑 Web Dynpro 组件如图 5-16 所示。

Object Name	Description
ZWD_ADOBE_INTERACTIVE	个人信息查询
COMPONENTCONTROLLER	个人信息查询----组件控制器
Component Interface	
INTERFACECONTROLLER	个人信息查询----组件控制器
Interface Views	
W_ADOBE_INTERACTI	个人信息查询----窗体
Views	
V_MAIN	个人信息查询----视图
Windows	
W_ADOBE_INTERACTIVE	个人信息查询----窗体

图 5-16

2）在 ABAP 工作台中的编辑界面编辑 Web Dynpro 视图。

选中左侧树形菜单中的视图 V_MAIN，选择“Context”选项卡，创建 Context 及其属性如图 5-17 所示。

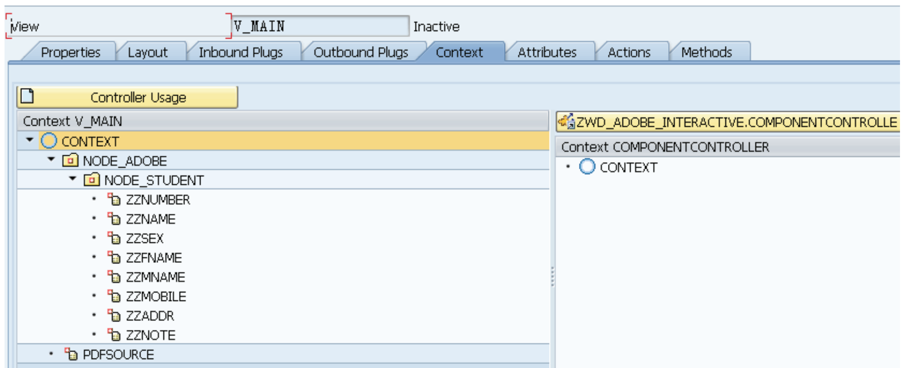


图 5-17

节点 NODE_ADOBE 设定如图 5-18 所示。

Change Node

Node Name	NODE_ADOBE
Interface Node	No
Input Element (Ext.)	No
Dictionary structure	
Cardinality	1..1
Selection	0..1
Init. Lead Selection	Yes
Singleton	No
Supply Function	
Mapping Path	

Add Attribute from Structure

图 5-18

节点 NODE_STUDENT 设定如图 5-19 所示。

Create Node

Node Name	NODE_STUDENT
Interface Node	No
Input Element (Ext.)	No
Dictionary structure	ZMNUMBER_T
Cardinality	1..1
Selection	0..1
Init. Lead Selection	Yes
Singleton	No
Supply Function	

Add Attribute from Structure | Additional Node

图 5-19

属性 PDFSOURCE 设定如图 5-20 所示。

Create Attribute

Attribute Name	PDFSOURCE
Type assignment	Type
Type	XSTRING
Read-only	No
Default Value	
Input Help Mode	Automatic
Ermittelte Wertehilfe	
Type of Input Help	

Additional Attribute

图 5-20

选中“Layout”选项卡，创建 UI 元素，添加按钮及 Interactive Form 元素，编辑布局如图 5-21 所示。

```
• CONTEXT_MENUS
▾ ROOTUIELEMENTCONTAINER
  ▾ GRP_STUDENT
      • CAPTION  [Header]
      • ZZNUMBER_LBL
      • ZZNUMBER
      • ZZNAME_LBL
      • ZZNAME
      • BTN_SEARCH
  ▾ GRP_STUDENT_FORM
      • CAPTION_1  [Header]
      • IAF_STUDENT
```

图 5-21

布局设定如图 5-22 所示。

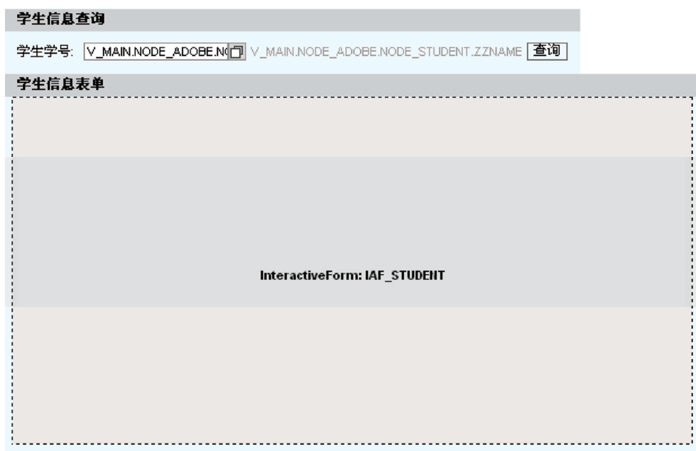


图 5-22

其中查询条件部分 UI 元素绑定如图 5-23 所示。

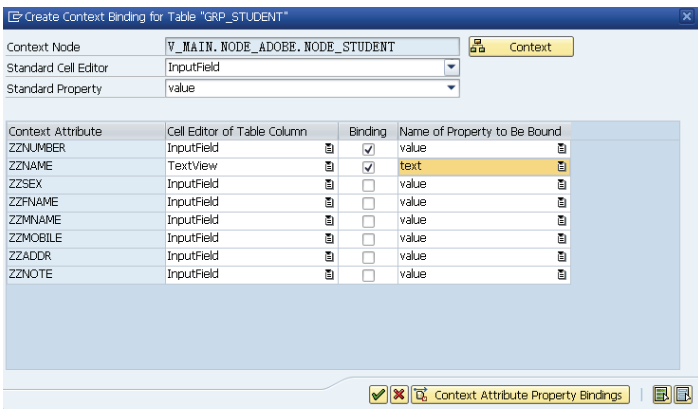


图 5-23

为按钮添加动作如图 5-24 所示。

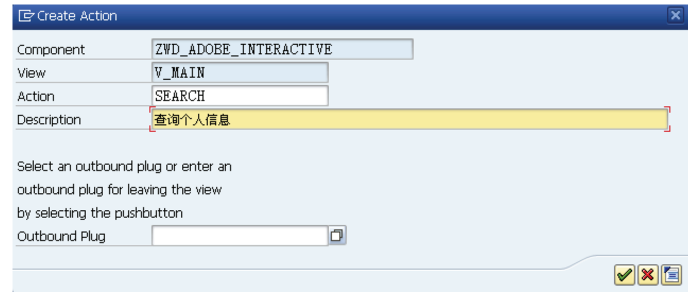


图 5-24

动作对应处理程序编辑如下：

```
METHOD ONACTIONSEARCH .
*     Context 存取用变量
      DATA LO_ND_NODE_STUDENT TYPE REF TO IF_WD_CONTEXT_NODE.

      DATA LO_EL_NODE_STUDENT TYPE REF TO IF_WD_CONTEXT_ELEMENT.
      DATA LS_NODE_STUDENT TYPE WD_THIS->ELEMENT_NODE_STUDENT.
      DATA LV_ZZNUMBER TYPE WD_THIS->ELEMENT_NODE_STUDENT-ZZNUMBER.

*    navigate from <CONTEXT> to <NODE_STUDENT> via lead selection
 LO_ND_NODE_STUDENT = WD_CONTEXT->PATH_GET_NODE( PATH = `NODE_ADOBE.
NODE_STUDENT` ).
* @TODO handle non existant child
*  IF lo_nd_node_student IS INITIAL.
*  ENDIF.

*  get element via lead selection
  LO_EL_NODE_STUDENT = LO_ND_NODE_STUDENT->GET_ELEMENT( ).
* @TODO handle not set lead selection
   IF LO_EL_NODE_STUDENT IS INITIAL.
   ENDIF.

*  取得输入值
LO_EL_NODE_STUDENT->GET_ATTRIBUTE(
   EXPORTING
     NAME =  `ZZNUMBER`
   IMPORTING
     VALUE = LV_ZZNUMBER ).

* navigate from <CONTEXT> to <NODE_STUDENT> via lead selection
 LO_ND_NODE_STUDENT = WD_CONTEXT->PATH_GET_NODE( PATH = `NODE_ADOBE.NODE_
STUDENT` ).
* @TODO handle non existant child
* IF lo_nd_node_student IS INITIAL.
* ENDIF.

* get element via lead selection
  LO_EL_NODE_STUDENT = LO_ND_NODE_STUDENT->GET_ELEMENT( ).

* @TODO handle not set lead selection
   IF LO_EL_NODE_STUDENT IS INITIAL.
   ENDIF.

* 取得个人信息
  SELECT    SINGLE   * FROM  ZMNUMBER_T
        INTO CORRESPONDING FIELDS OF LS_NODE_STUDENT
             WHERE  ZZNUMBER  = LV_ZZNUMBER .
```

```
    * Context 赋值
    LO_EL_NODE_STUDENT->SET_STATIC_ATTRIBUTES(
      STATIC_ATTRIBUTES = LS_NODE_STUDENT ).
ENDMETHOD.
```

3）在 ABAP 工作台中的编辑界面编辑 Adobe Form。

Interactive Form 元素属性设定如图 5-25 所示。

双击 templateSource 属性，创建 Form 并设定接口如图 5-26 所示。

Property	Value	Bin...
Properties (InteractiveForm)		
ID	IAF_STUDENT	
additionalArchives		
contextMenuBe...	Inherit	
contextMenuId		
dataSource	V_MAIN.NODE_ADOBE	
displayType	native	
enabled		
height	600px	
pdfSource	V_MAIN.PDFSOURCE	
readOnly		
templateSource	ZTEST_INTERACTIVE	
tooltip		
visible	Visible	
width	600px	
Events		
onSubmit		
Layout Data (FlowData)		
cellDesign	padless	
vGutter	None	

图 5-25

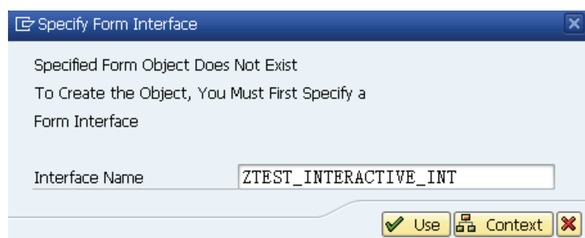


图 5-26

单击图 5-26 中的 Context 按钮，指定 Form 映射的 Context 如图 5-27 所示。

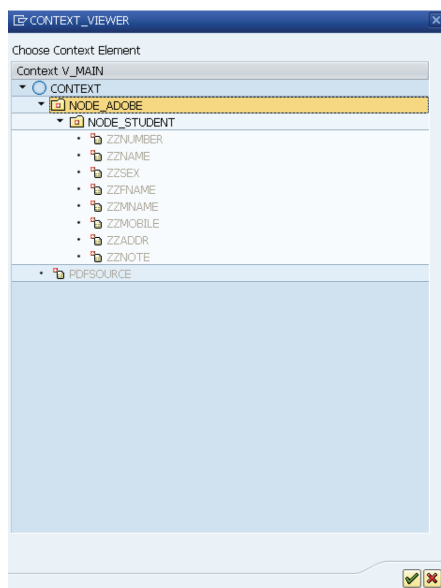


图 5-27

单击按钮保存当前页面，编辑 Adobe Form 属性如图 5-28 所示。

图 5-28

编辑 Adobe Form 布局如图 5-29 所示。

在 Form 编辑页面上，按照以下路径将 Web 屏幕脚本插入 Form，如图 5-30 所示。

图 5-29

图 5-30

编辑 FormCalc 脚本如图 5-31 所示。

```
NODE_ADOBE.#subform[0].Table1.Row3.ZZSEX::initialize - (FormCalc, client)
if ($record.NODE_STUDENT.ZZSEX.value == "M") then
NODE_ADOBE.#subform[0].Table1.Row3.ZZSEX.value.#text  = "男";
elseif ($record.NODE_STUDENT.ZZSEX.value == "W") then
NODE_ADOBE.#subform[0].Table1.Row3.ZZSEX.value.#text  = "女";
endif
```

图 5-31

编辑 Adobe Form，如图 5-32 所示。

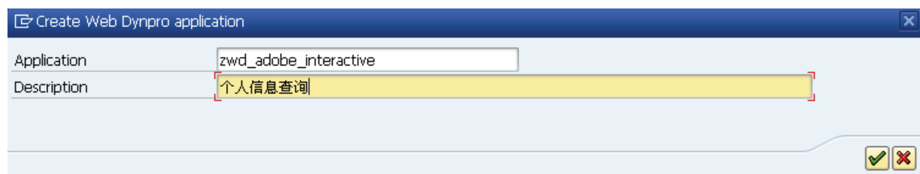


图 5-32

4）在 ABAP 工作台中的编辑界面测试 Adobe Form，如图 5-33 所示。

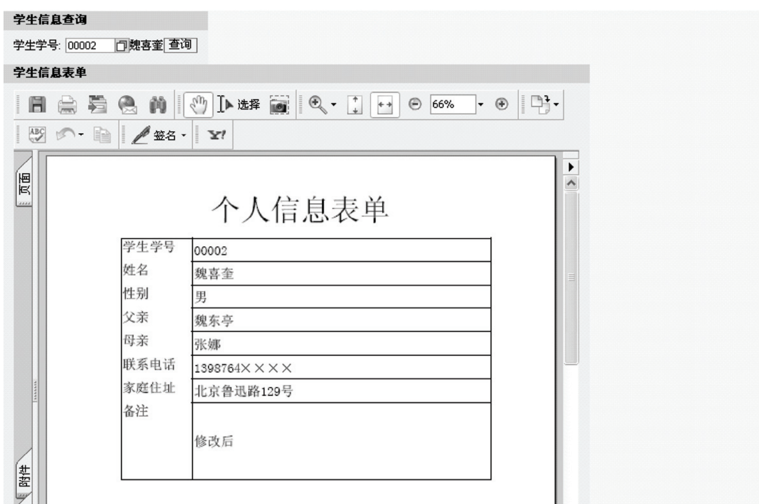


图 5-33

要为数据视图的表单提供新的元素，首先需要增强相关的 Web Dynpro 的 Context。可以通过 Web Dynpro 资源管理器的 Context 编辑器进行编辑，如图 5-34 所示。通过双击视图上的 InteractiveForm UI 元素可以转到表单生成器。此时 XML 架构连同表单生成器中的数据视图会自动调整，如图 5-35 所示。

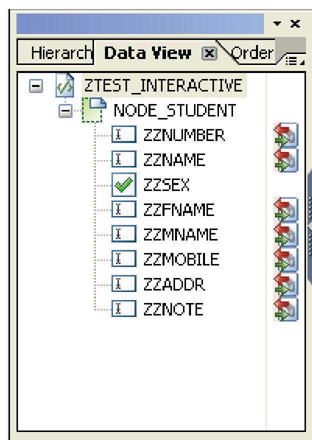


图 5-34

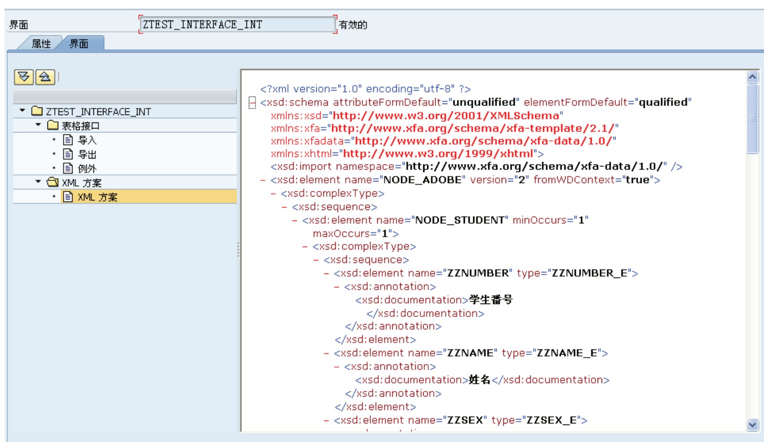


图 5-35

注：如果使用 templateSource 整合现有的表单和它的接口不是 XML 模式，在其后就不可以添加任何多余数据字段。

5.4 插入已有 PDF 表单（MIME 对象）

存储在系统中的 MIME 库中的 MIME 类型为 application/pdf 的 MIME 对象，它可以集成到 Web Dynpro 应用程序中。

（1）前提

要存储表单到 MIME 库中，最简单的方法是通过以下路径导入表单到 Web Dynpro 组件中，如图 5-36 所示。

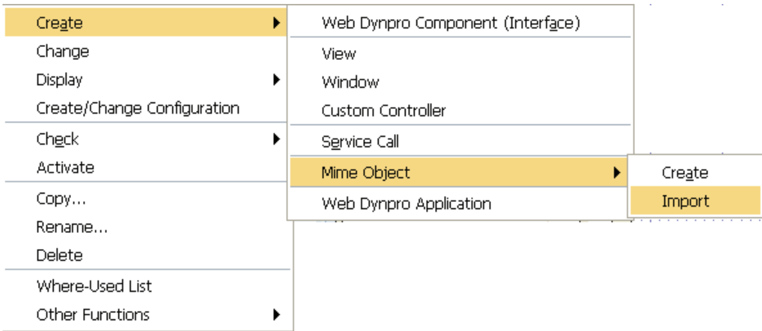


图 5-36

导入的 PDF 会在组件的对象列表中的 MIMEs 下列出，如图 5-37 所示。

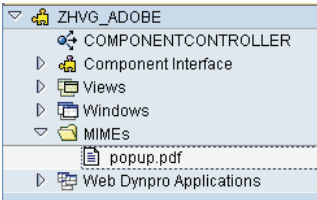


图 5-37

（2）操作（Procedure）

1）在 Web Dynpro 资源管理器中，创建一个视图用以添加表单。在视图 Context 中，创建类型为 XSTRING 的 Context 属性用于绑定 InteractiveForm UI 元素的 pdfSource 属性。在视图布局设计器里，拖动 InteractiveForm UI 元素到布局中并做相关属性绑定。

2）转到视图 Methods 标签页，编辑函数 WDDOINIT 如下：

```
METHOD WDDOINIT .
  DATA:
      ELEM_CONTEXT           TYPE REF TO IF_WD_CONTEXT_ELEMENT,
      STRU_CONTEXT           TYPE IF_VIEW=>ELEMENT_CONTEXT ,
      ITEM_STATIC_PDF        LIKE STRU_CONTEXT-STATIC_PDF.
```

```
*        get element via lead selection
     ELEM_CONTEXT = WD_CONTEXT->GET_ELEMENT(  ).
     DATA: MR       TYPE REF TO IF_MR_API,
                LURL    TYPE        STRING.

*     get name of component
     DATA: L_COMPONENT TYPE REF TO IF_WD_COMPONENT,
               L_COMPONENT_INFO TYPE REF TO IF_WD_RR_COMPONENT,
               L_NAME TYPE STRING.

     L_COMPONENT = WD_COMP_CONTROLLER->WD_GET_API( ).
     L_COMPONENT_INFO = L_COMPONENT->GET_COMPONENT_INFO( ).
     L_NAME = L_COMPONENT_INFO->GET_NAME( ).
*    get mime object
     MR = CL_MIME_REPOSITORY_API=>GET_API( ).
    CALL METHOD CL_WD_UTILITIES=>CONSTRUCT_WD_URL( EXPORTING APPLICATION_
NAME = L_NAME
                                                IMPORTING OUT_LOCAL_URL     = LURL ).
  CONCATENATE LURL '/' 'popup.pdf' INTO LURL.

  MR->GET( EXPORTING  I_URL      = LURL
                 IMPORTING  E_CONTENT = ITEM_STATIC_PDF  ).
  ELEM_CONTEXT->SET_ATTRIBUTE(
              EXPORTING
                  NAME =  `STATIC_PDF`
                  VALUE = ITEM_STATIC_PDF ).

ENDMETHOD.
```

在该例子中只需要指定 MIME 对象的名称：popup.pdf。

还可以通过手动插入路径接收 MIME 对象：

```
METHOD WDDOINIT .
     DATA: MR       TYPE REF TO IF_MR_API,
                  CONTENT TYPE        XSTRING.

     MR = CL_MIME_REPOSITORY_API=>GET_API( ).
     MR->GET( EXPORTING  I_URL      = 'sap/bc/webdynpro/sap/zhvg_adobe2/popup.pdf'
                   IMPORTING  E_CONTENT = CONTENT ).
*      set mime object
       wd_context->set_attribute( name = 'PDFSOURCE' value = content ).
ENDMETHOD.
```

3）保存、编译、执行，查看运行结果。

5.5 表单集成中的数字签名（Digital Signatures）

编辑表格时，其内容安全是关键。使用数字签名是一个好方法，使原始的表单的真伪得以验证。

（1）前提

1）ADS 连接。对于签名，你需要通过 SSL 或 HTTPS 的一个安全连接。对于这一点，系统需要一个 HTTP 类型的 RFC 连接，该 RFC 连接到外部服务器的 ADS。外部 ADS 服务必须是以 J2EE 引擎服务器的服务。

2）附有签名栏的文档。设置和检查签名，表单必须有一个签名栏。可以使用表单生成器在布局中插入签名栏（事务代码：SFP）。

签名是否是服务器端或客户端没有任何区别。

（2）在表单上签字

1）创建一个带有数字签名的表单，在表单上进行数字签名时，单击表单上的签名栏，如图 5-38 所示。

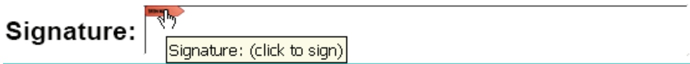


图 5-38

2）确认随后的提示，将签名信息输入签名字段中，如图 5-39 所示。示例：



图 5-39

（3）检查签名

通过插入表单的 Send 按钮验证签名的有效性检查。该按钮用于后台间签名的发送。

接口 IF_FP_PDF_OBJECT 的方法 GET_SIGNATURES 用于此目的。

5.6 支持 Adobe 库中的 UI 元素

在 Web Dynpro ABAP 应用程序的 Adobe 设计表单开发时，进入到表单设计器的“布局”选项卡中，利用这些 UI 元素编辑表单。根据表单集成的类型，以下库中的元素可用于 Web Dynpro ABAP 中，如图 5-40 所示。

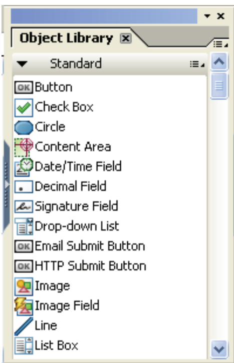


图 5-40

1. ZCI 表单

如果在表单生成器中选择了 ZCI 布局作为布局类型属性，则作为一个基于 ZCI 表单的开发者，可以使用 Web Dynpro 原生库。目前 Web Dynpro ABAP 环境下所支持的 Adobe 的 UI 元素如下：

（1）Send 按钮（SubmitToSAP）

使用 Web Dynpro 表单的 UI 元素 SubmitToSAP，发送一个 PDF

表单的输入数据到 SAP 后台系统。与 Web Dynpro 的输入数据同步，绑定到 InteractiveForm 的 Web Dynpro 的 Context 节点（指在视图设计器中绑定到属性 DataSource 的节点属性）会自动更新。Web Dynpro UI 元素 InteractiveForm 的动作 onsubmit 也被触发。

（2）EnumeratedDropDownList 和 EnumeratedDropDownListNoSelect

在表单上这两个 UI 元素用于可检索对象，在 Adobe 设计器中使用拖放 Web Dynpro 库的 UI 元素编辑表单。两个元素之间的差异是，如果在所有的数据部分中没有条目或当前选择列表没有相关条目，则 EnumeratedDropDownListNoSelect 不会自动选择一个值。EnumeratedDropDownList 会自动检索选择列表中的第一项。

（3）操作

1）将 Adobe 的 UI 元素拖放至模板，并把它绑定到数据视图中的 Context。

2）改变元素值绑定。

- 选择 Element Values。在下面弹出的 Objects 的下方会出现组标题 Binding：

```
$ record.sap-vhlist.REPLACE_THIS.item[*]
```

- SOM 的数据绑定表达式替换 REPLACE_THIS。

从 Objects 菜单中可以得到这个值。该值以$record 开头，如绑定节点 Node.DVH3。

正确的值的示例：$ record.sap-vhlist.Node\.DVH3.item[*]

注：EnumeratedDropDownList 的值采取从 Web Dynpro 属性 VALUE_SET 绑定时。如果该类型的属性是 DDIC（ABAP 字典）和定义域的固定值，则这些值会被显示出来。也可以对 VALUE_SET 编程，以填补在运行时的值。

（4）输入帮助

Adobe 的 UI 元素输入帮助，使得它可以用帮助的形式来调用输入，如图 5-41 所示。



图 5-41

需要注意的是，XFO 编码的输入帮助必须调整。输入绑定的属性字段的名称（在本例中是 CARRID）。

1）初始化时赋值：

```
var fieldName = "**INSERT_NAME_HERE**";
```

2）输入时赋值，如图 5-42 所示。

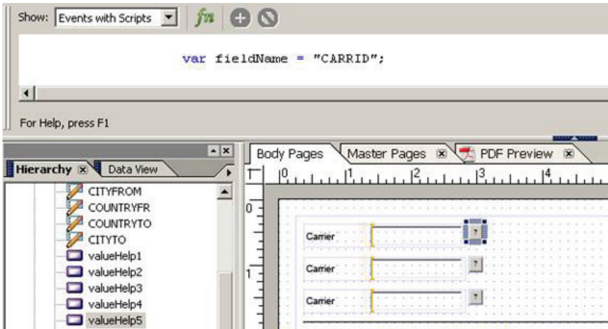


图 5-42

因此，Adobe 的 UI 元素的输入帮助是连接到 Web Dynpro ABAP 输入帮助的。

2. ACF 表单

ACF 的基础上开发的表单是基于 Web Dynpro ActiveXlibrary（请参阅关于 PDF 的打印表格的文档）的。

目前 Web Dynpro ABAP 环境下支持的元素：Send。

使用 Web Dynpro 表单 UI 元素 Send，是将 PDF 表单中输入的数据发送到用于存储的 SAP 后台系统。UI 元素也提供了一个自动生成的事件。

5.7 交互式表单应用

有很多原因，Web Dynpro 应用程序需要输出表单。大多数嵌入式的表单用于打印和存档，因此不提供互动式的表单。但 Web Dynpro 框架仍然可以使用内嵌的 PDF 表单用于交互。使用 PDF 作为 Web Dynpro 应用程序输入的模板可能是特别有用的，尤其是当用户熟悉表单的外观及内容的情况下，这种表单在系统中是非常有用的。

在系统的默认设定中，如果 UI 元素的类型 InteractiveForm 是非交互式的，那就是 enabled 属性的复选框未选中。如果要使用交互报表，则必须选中 enabled 属性复选框。

注：如果希望应用程序使用交互式报表，则需要选中该复选框，即表示该表单作为 Active Components Framework（ACF）用于呈现在客户端（一般用户浏览器）中，因此 ACF 必须安装到客户端。通常情况下相关的应用程序第一次被调用时，ACF 会从 SAP NetWeaver AS 中集中安装到客户浏览器中。然而，一个用户可能没有足够的 ACF-CAB 文件安装在本地的授权也是可能的。在这种情况下，用户不能显示表单。如果表单 enabled 属性复选框没有被选中，则不需要进行 ACF 显示。

5.8 基于 ABAP 数据字典接口表单

前面对基于 XML 的接口进行了介绍，与基于 ABAP 字典接口的表单的建立在集成过程中是相同的。然而，这样的接口不容易链接到 Web Dynpro 组件的 Context 中。Web Dynpro 内创建的所有表单不会自动配备一个基于 XML 的接口。然而，在某些情况下，Web Dynpro 的 Context 是有必要集成到 Web Dynpro 应用程序中的一个基于功能模块接口的表单的。Web Dynpro 运行时，原则上使得它可以在 Web Dynpro 应用程序内显示或打印这样一种表单。在某些特殊情况下，它甚至有可能在交互模式下运行，该类型也是一种表单的集成形式。

注：一个基于功能模块接口的表单只能用 PDF 打印。

UI 元素 InteractiveForm 也可以限制输入。由于用户输入的值都没有传递到各自的属性。表单显示在改变模式下，用户完成输入，它被作为一个整体存储在 Context 属性中，该属性映射到属性 pdfSource 上。这对打印或归档表单是特别有用的。在这种情况下，输入的验证在本框架下还没有提供。

注：一旦窗体设置为输入启用，该属性将被保留。它不能再被重置为一个纯粹的显示模式。

UI 元素 InteractiveForm 的接口类中包含方法 SET_LEGACY_EDITING_ENABLED 可以动态设置输入的启用，方法可以传递一个布尔参数用于设置输入。当表单被设置如下时输入

功能被启用。

- 参数的值为 X。
- UI 元素 InteractiveForm 的 enabled 属性被选择。
- 属性 pdfSource 被绑定到一个类型为 XSTRING 的 Context 属性上。
- 属性 dataSource 被绑定到一个与表单 Context 兼容的 Context 属性上。

表单使用 Context 在 InteractiveForm 和 Web Dynpro 的 Context 之间交换数据。如果表单是基于一个 ABAP 数据字典接口的，则数据从接口传递到表单 Context。程序员可以选择更改属性名称。但在这种情况下，必须使用指定的属性名。

下面的代码片段显示了如何配置为输入的表单：

```
METHOD WDDOMODIFYVIEW.
    DATA: LR_INTERACTIVE_FORM TYPE REF TO CL_WD_INTERACTIVE_FORM,
    LR_METHOD_HANDLER TYPE REF TO IF_WD_IACTIVE_FORM_METHOD_HNDL.
    CHECK FIRST_TIME = ABAP_TRUE.
    LR_INTERACTIVE_FORM ?= VIEW->GET_ELEMENT('INTERACTIVE_FORM_1').
    LR_METHOD_HANDLER ?= LR_INTERACTIVE_FORM->_METHOD_HANDLER.
    LR_METHOD_HANDLER->SET_LEGACY_EDITING_ENABLED( ABAP_TRUE ).
    ...
ENDMETHOD.                              "WDDOMODIFYVIEW
```

5.9 隐藏 Adobe Toolbar

可以设置隐藏 Adobe Reader 的工具栏。在默认设置中，该工具栏是显示的，如图 5-43 所示。

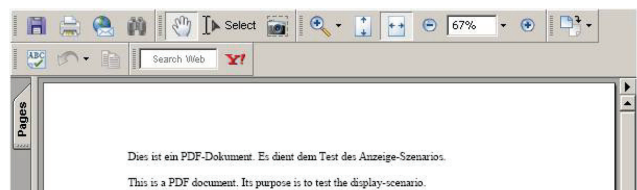


图 5-43

可以使用 UI 元素的接口 InteractiveForm IF_WD_IACTIVE_FORM_METHOD_HNDL 的方法 SET_HIDE_TOOLBARS 隐藏工具栏，如图 5-44 所示。

```
lr_method_handler->set_hide_toolbars( abap_true );
```

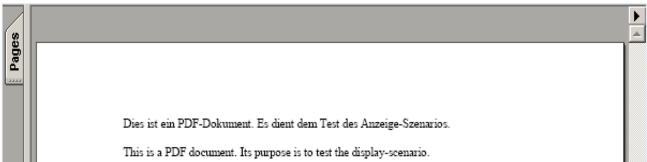


图 5-44

用户可以通过选择应用程序中的功能键 F8 来隐藏（或显示）工具栏。

第 6 章　ALV 在 Web Dynpro ABAP 中的应用

SAP List Viewer（ALV）是一种灵活的工具，用于显示列表和表结构。作为标准功能，该工具提供了常见的列表操作，可以提高自定义选项。这允许用户在比较大的应用程序中使用 ALV。

对于最终用户来说，标准的输出包括工具栏、标题及输出表。用户可以通过使用特定的对话框进行设置，从而更改列的显示，进行更复杂的排序、汇总。作为应用程序开发人员，在 ALV 输出时有多种选项来定义的外观、功能和运行时的动作。

程序员还可以使用 ALV 组件。与其他所有预定义的组件一样，可以将 ALV 组件添加到自己的应用程序组件中。该应用组件可以使用 ALV 组件具有的所有功能，这样应用程序为用户提供了强大的工具来显示列表。

ALV 组件部分基于 Web Dynpro 的 UI 元素 Table。这意味着，许多 Table 元素的属性在 ALV 组件中是可用的，这包括使用不同的单元格编辑器、背景颜色和尺寸规格。除此之外，下面的选项也是可用：

- 指定的列包含在用户的列集，用户输出 ALV 时这些都是可见的。
- 可以排序和过滤 ALV 输出的值，并执行计算。
- 决定是否在那些范围内编辑 ALV 输出。
- 指定在列的单元格中显示何种 UI 元素的值。
- 在工具栏中提供 UI 元素，用户可以执行特定的应用程序的功能。
- 可以指定是否及如何处理用户交互的 ALV 输出。
- 用户可以保存不同显示的设置。
- 可以配置 ALV 输出的上方和下方的特殊区域。

6.1　在应用程序中集成 ALV

1．目的

如果想在应用程序中使用 ALV 输出，则根据所需的结果，必须遵循相应的步骤。下列变式列出了最重要的方法，可以用于调用 ALV：

1）简单的调用（无 ALV 配置）：仅显示一个 ALV 输出。不想进行任何更改。

2）简单的调用（有 ALV 配置）：要更改使用应用程序的标准输出。要做到这一点，需要 ALV 配置。

3）外部视图的 ALV 输出（无 ALV 配置）：ALV 输出仅仅是在一个视图中的许多元素之一。用户不想改变 ALV 输出，并不需要 ALV 配置。

4）外部视图的 ALV 输出（有 ALV 配置）：ALV 输出仅仅是在一个视图中的许多元素之一。用户要使用应用程序更改标准输出。要做到这一点，需要 ALV 配置。

2．前提

程序员必须根据上面列出的所有变式进行如下（ALV 无关）操作：

1）为应用程序创建 Web Dynpro 组件，将其保存激活。

2）创建一个 Context 节点，基数为 0.. N。该 Context 节点的结构（名称和数据类型）在下一步骤中，连接到该 Context 节点的内部数据表的结构及属性是相同的。

3）将内部数据表数据绑定到 Context 节点（如使用供给方法）。要做到这一点，可以使用接口 IF_WD_CONTEXT_NODE 中的方法 BIND_TABLE。

4）在应用程序的 ALV 接口控制器中提供的数据 Context 节点上使用外部数据映射（将创建的 Context 节点映射到 ALV 接口控制器中提供的数据 Context 节点）。

3．操作流程

若上面列出的先决条件都满足，则过程操作取决于所需的结果。

（1）简单的调用（无 ALV 配置）

1）在应用程序组件中为 ALV 组件 SALV_WD_TABLE 定义一个组件使用。在操作时，可以指定组件使用的名称（如 MY_ALV）。

2）在应用程序数据和 ALV 接口控制器提供的 DATA Context 节点建立外部 Context 映射。

3）编辑组件的窗体，嵌入 ALV 组件 SALV_WD_TABLE 的接口视图。

（2）简单的调用（有 ALV 配置）

1）在应用程序组件中为 ALV 组件 SALV_WD_TABLE 定义一个组件使用。在操作时，可以指定组件使用的名称（如 MY_ALV）。

在组件控制器中定义该组件使用。因为需要进行 ALV 配置，所以选择控制器访问变式（组件接口）。

2）编辑组件的窗体，嵌入 ALV 组件 SALV_WD_TABLE 的接口视图。

3）在应用程序数据和 ALV 接口控制器提供的 DATA Context 节点建立外部 Context 映射。

4）ALV 输出配置，必须遵循以下两个额外的步骤：

- 在组件的控制器（如 WDDOINIT）的方法中实例化所使用的 ALV 组件。
- 会收到 ALV 配置模型和对象模型，以及字段对象和列对象。

（3）外部视图的 ALV 输出（无 ALV 配置）

1）决定在组件控制器中或在应用程序视图中的 Context 下是否需要 Context 节点。如果有必要，则在视图中创建适当的 Context 节点和映射到它的组件控制器 Context 的节点。

2）在应用程序组件中为 ALV 组件 SALV_WD_TABLE 定义一个组件使用。在操作时，可以指定组件使用的名称（如 MY_ALV）。也可以在视图属性中定义这个组件的使用。既然不需要 ALV 配置，就不用选择控制器访问的变式（没有组件接口）。

3）在应用程序数据和 ALV 接口控制器提供的 DATA Context 节点建立外部 Context 映射。

4）在视图布局中所需的位置创建 UI 元素 ViewContainerUIElement。

5）编辑组件的窗体，嵌入以创建视图，在视图的 UI 元素 ViewContainerUIElement 下嵌入 ALV 组件 SALV_WD_TABLE 的接口视图。

（4）外部视图的 ALV 输出（有 ALV 配置）

1）决定在组件控制器中或在应用程序视图中的 Context 下是否需要 Context 节点。如果有必要，则在视图中创建适当的 Context 节点和映射到它的组件控制器 Context 的节点。

2）在应用程序组件中为 ALV 组件 SALV_WD_TABLE 定义一个组件使用。在操作时，可以指定组件使用的名称（如 MY_ALV）。也可以在视图属性中定义这个组件的使用。既然需要 ALV 配置，就需要选择控制器访问的变式（组件接口）。

3）在视图布局中所需的位置创建 UI 元素 ViewContainerUIElement。

4）编辑组件的窗体，嵌入以创建视图，在视图的 UI 元素 ViewContainerUIElement 下嵌入 ALV 组件 SALV_WD_TABLE 的接口视图。

5）ALV 输出配置，必须遵循以下两个额外的步骤：

- 在组件的控制器（如 WDDOINIT）的方法中实例化所使用的 ALV 组件。
- 会收到 ALV 配置模型和对象模型，以及字段对象和列对象。

6.1.1 组件使用定义

在应用程序中，要使用组件SALV_WD_TABLE进行ALV输出，可以将这种ALV组件作为外部组件集成到Web Dynpro组件。要做到这一点，可以定义组件的使用。

注：如果想显示多个不同 ALV 输出，则各自不同的组件使用唯一的名称可作为这些输出之间的区分。

如何定义组件的使用取决于想如何使用 ALV 组件。

- 如果在屏幕上 ALV 的输出是唯一的元素，那么 ALV 组件的使用只要在应用程序组件中定义（在“组件编辑器”层次结构中顶级节点的组件）即可。这是最简单的一种情况。ALV 显示出来后才可以配置输出。
- 如果在屏幕上 ALV 输出是唯一的元素（同上），但想改变外观、功能或运行时的动作，则在应用程序的组件控制器的属性中指定组件的使用。选择控制器访问变式。
- 如果要插入 ALV 输出到一个复杂的布局中，则在应用程序的视图属性中，定义组件使用，该视图作为 ALV 输出要显示的视图（而不是组件控制器）。

根据是否要使用标准的 ALV 输出显示，或是否要改变它，选择无控制器访问或控制器访问的变式。

6.1.2 供应数据

要提供想要的数据用来 ALV 输出，可以执行以下 3 个步骤。

1）内部数据表：定义一个内部表。

2）Context 节点：在应用程序中定义一个合适的 Context 节点及其属性。

3）将数据传输到 ALV 组件：可以为 ALV 组件定义 Context 节点。

1．内部数据表

内部数据表中的数据必须是扁平结构、可用的。这是唯一的先决条件。做到这一点最简单的方法是使用现有的 DDIC 结构。

2．Context 节点

ALV 组件需要一个 Context 节点用于存储内部数据表中的数据。该 Context 节点具有以下属性：

- 它的基数为 0.. N。
- 它仅包含属性没有子节点：对应于内部数据表中的每一列，需创建一个相同名称和数据类型的属性。
- 所有的属性都是静态的，不是动态的。

如果内部数据表是基于一个 DDIC 结构，则使用其创建 Context 节点及其属性是最简单的方法。

内部数据表中的数据传输到应用程序 Context 节点，即该节点到数据表的连接。要做到这一点，使用方法 BIND_TABLE，如在控制器的供给方法中使用。

3．将数据传送到 ALV 组件

在前面的步骤中，创建了 ALV 输出用的 Context 节点。现在，必须把这个传送到 ALV 组件。可以通过以下方式来做到这一点：

- 使用外部 Context 映射。
- 使用方法 SET_DATA。

注：因为程序员可能定义了不同的组件使用的各种不同的组件（或不同的 ALV 组件），所以必须指定确切的 ALV 组件使用，其 Context 节点的数据包含 Context 映射。

（1）外部 Context 映射

ALV 组件包含一个单独的名为 DATA 的 Context 节点。该节点包含 ALV 输出的结构和数据，能够满足显示所需的输出。最简单的方式是使用外部 Context 映射将数据传输到数据 Context 节点。

（2）方法 SET_DATA

通过使用 SET_DATA 方法（引用类型 IF_WD_CONTEXT_NODE）也可以将数据传送到 ALV 组件的 Context 节点。可以这样做，如在 WDDOINIT 方法调用该方法。

注：如果要以另一结构在稍后进行 ALV 输出时，也可以使用方法 SET_DATA。然而，在这种情况下，不能在方法 WDDOINIT 或其他标准方法中使用该方法。如果 ALV 输出已经显示一次，则只有在 ALV 组件的事件处理程序中定义该数据（调用 SET_DATA 方法）。

方法 SET_DATA 总是删除可能存在的任何的 ALV 配置模型。如果想配置 ALV 的输出，一旦 SET_DATA 方法被调用，则程序员必须重新得到 ALV 配置模型。

（3）在 ALV 组件中使用数据

在 ALV 组件的 Context 中，Context 节点的 DATA 仅包含应用程序相应的 Context 节点的引用。这反过来是内部数据表的一个引用。

默认情况下，ALV 组件使用引用的数据，通过各种机制对数据进行过滤器，计算和其他标准的 ALV 功能，改变 ALV 输出，然后显示在屏幕上。

（4）重要的例外：排序（Sorting）

这里 ALV 使用整个数据集的数据，所以记录可以重新排序。为此，ALV 组件暂时控制内部数据表，在此期间，应用程序相应的 Context 节点失效。这确保了应用程序不能访问 Context 节点，而编辑 ALV 组件内部数据表的数据。

一旦内部数据表已经重新排序，则 ALV 重建 Context 节点，再次释放它的应用程序，并

显示相应的数据。

注：这样做确保了内部数据表中的数据从来没有被复制。这是很重要的，因为大量的数据将大大影响性能和内存空间。

当程序员设计应用程序时，请注意这一机制的副作用：

- 当 Context 节点失效时，当前选择的信息，特别是 lead selection 选择的信息将丢失。
- 如果应用程序 Context 节点中有子节点（master-detail 方案），则当 ALV 组件 Context 节点失效时这些子节点信息也将丢失。如果应用程序试图访问子节点，则运行时发生错误。

6.1.3 使用 ALV 视图

ALV 组件中包含以下视图，可以根据需要使用相关视图：

1）表格视图。表视格图是 ALV 的中央视图。要显示 ALV 的输出，就需要这个视图。

注：表 UI 元素所在视图显示在 ALV 输出的容器中。这是 ALV 组件的一个基本组成部分。因此 ALV 的布局不能改变。

2）服务视图。

该视图中包含“设置”对话框，用户可以更改列显示的设置、排序、筛选等。

默认情况下，当用户选择工具栏上的“设置”超链接时，对话框在 ALV 输出上面弹出。如果想在屏幕上的其他地方显示对话框，则也可以使用该视图。

如果想在应用程序中显示 ALV 输出，则需要表格视图。

可以将表格视图以不同的方式插入到应用程序中：

- 将其直接插入应用程序的窗体中。在这种情况下，不能在屏幕上显示任何其他元素，除了 ALV 输出。
- 将其插入到应用程序中的视图集（Web Dynpro Java 的概念）的单元格。
- 在视图需要的位置创建一个类型为 ViewContainerUIElement 的 UI 元素，然后将 ALV 表格插入这个 UI 元素中。

6.1.4 ALV 配置模型对象

配置 ALV 输出主要有以下几个主要方面：

（1）表格设置

表 UI 元素的数据包括应用程序结构以及结构内各字段的技术设置数据，从而确定 ALV 输出的外观或功能。也可以定义，如程序员可以定义一个简单的二维表或定义某一列作为与领先的层次列的层次结构列表作为 ALV 输出的对象。

（2）字段设置

字段描述数据用于 ALV 输出。字段名称对应于 Context 节点的属性名称，所有的字段对象都是参照指定的 Context 节点自动生成的。因此，在 ALV 配置模型中的字段与 Context 节点的每一个属性都具有相同的名称。

连接内部数据表和 Context 节点是实现使用 ALV 启动应用程序的所有先决条件。可以对

应用程序的报表中的数据进行排序、过滤或执行应用程序。所有这些功能（ALV 服务）本质上是字段对象的方法。但是，字段的默认状态为显示。要想对某一字段做到隐藏，需要对列进行设置。

（3）列设置

列对象是可见的元素，它定义 ALV 输出。列具有与 Context 节点属性和相应字段对象相同的名称。列对象中包含是否及如何将字段名称相同的数据在 ALV 输出的设置。如果不想显示字段的值，则可以在程序中删除相应的列对象。

可以使用列设置来管理列表中的所有列对象（表 UI 元素的列）。

（4）标准功能设置

ALV 提供了许多功能。下面列出了这些标准的 ALV 功能中最重要的几项。

- ALV 服务：排序、过滤、聚合（计算），以及设置这些服务的选项。
- 可以在设置中编辑 ALV 输出，如插入和删除行。
- 导出 ALV 输出到 Microsoft Excel 或生成 PDF 格式的打印版本。

所有这些标准功能都可以使用相关的 UI 元素实现。可以使用标准功能设置隐藏或显示这些 UI 元素。

（5）应用程序特定的功能设置

可以根据需要在用户的应用程序中定义尽可能多的功能，并提供合适的 UI 元素，通过其用户就可以运行这些功能了。

作为应用程序开发人员，可以配置所有领域。如果必要，可以使用一个完整的面向对象 ABAP 编程的对象模型，可以用它来改变 ALV 的输出。每个领域是一个接口类。它们集成到类 CL_SALV_WD_CONFIG_TABLE 中。

6.1.5 ALV 配置模型

为了使 ALV 配置模型可用，可以以调用相关方法。例如，在钩子方法 WDDOINIT 中调用类 IWCI_SALV_WD_TABLE 的方法。可以选择下列方法之一：

1）GET_MODEL：系统将返回所有列对象和字段对象的完整的 ALV 配置模型。这是默认的情况。该方法不具有任何参数。

2）GET_MODEL_EXTENDED：系统将返回所有的字段对象（如上），但只有需要时返回列对象。

例如，如果要管理大量的字段对象，但只需要显示几列，则可以先创建字段对象，然后根据要显示列的内容创建列对象。

方法 GET_MODEL_EXTENDED 中的参数 S_PARAM（类型 IF_SALV_WD_TABLE=>S_TYPE_PARAM_GET_MODEL）包含 DEFAULT_COLUMNS 字段：这使程序员可以指定是否返回列对象。

注：在程序运行后，程序员可以指定一个新的内部数据表用于 ALV 输出。要做到这一点，使用方法 SET_DATA。使用这种方法，还可以自动删除整个 ALV 配置模型的所有字段对象和列对象。为了使配置模型应用于新的数据表中，应再次调用方法 GET_MODEL 或方法 GET_MODEL_EXTENDED。

6.1.6　类图（Class Diagram）

ALV 配置模型的对象模型由以下类和接口组成，如图 6-1 所示。

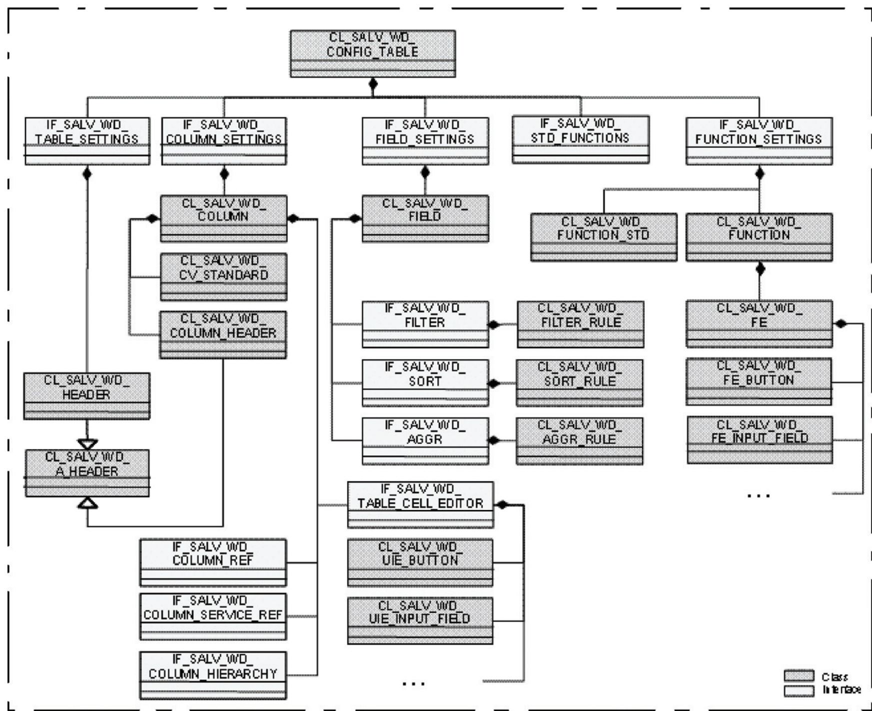


图 6-1

这个类图进行了简化。一些超级类在此图中已被删除。

6.2　管理 ALV 输出领域

ALV 输出由以下的不同区域组成，以在一定程度上各项彼此独立，可以对其创建、变更和删除。

- 页眉。
- ALV 标题。
- 字段和列。
- 列标题。
- 页脚和滚动条。
- 页脚区域（见页眉和页脚区域）。

用户可以使用不同的领域。

6.2.1　ALV 标题（ALV Output Header）

默认情况下，ALV 输出没有标题。程序员可以创建一个标题，并在 ALV 输出的上面显示。

标题可以有以下几部分组成：文本、图形、工具提示。

ALV 输出有标题的，可以进行以下设置：

- 创建、取得或删除标题对象。
- 设置标题文字。
- 设置标题图形路径。
- 设置标题图形位置。
- 设置标题文字工具提示。

（1）创建、取得或删除标题对象

ALV 输出的标题是一个实例的类 CL_SALV_WD_HEADER。要创建或删除对象，需使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-1。

表 6-1

功　　能	方　　法
创建标题对象	CREATE_HEADER
取得标题对象	GET_HEADER
删除标题对象	DELETE_HEADER

（2）设置标题文字

ALV 输出标题文字类型为 String。要定义标题，可以使用类 CL_SALV_WD_HEADER 中的方法，见表 6-2。

表 6-2

功　　能	方　　法
设置标题文字	SET_TEXT
取得标题文字	GET_TEXT

（3）设置标题图形路径

可以在 ALV 输出的标题区域显示任何图形，只要该图形兼容 Web Dynpro 环境。要定义图形文件的路径或 ID，使用类 CL_SALV_WD_HEADER 的方法，见表 6-3。

表 6-3

功　　能	方　　法
设定图形文件的路径或 ID	SET_IMAGE_SOURCE
取得图形文件的路径或 ID	GET_IMAGE_SOURCE

（4）设置标题图形位置

可以选择是否在标题中文字之前或之后显示图形。要做到这一点，使用类 CL_SALV_WD_HEADER 的方法。

表 6-4

功　　能	方　　法
标题内设定位置	SET_IMAGE_FIRST
标题内取得位置	GET_IMAGE_FIRST

（5）设置标题文字工具提示

当用户把光标移到 ALV 输出的标题时，标题头的工具提示变得可见。指定标题头的提示，使用类 CL_SALV_WD_HEADER 中的方法，见表 6-5。

表 6-5

功　　能	方　　法
设定工具提示文字	SET_TOOLTIP
取得工具提示文字	GET_TOOLTIP

6.2.2　字段

字段对象可用于以下两种情况：

1）在进行 ALV 输出前应用标准的 ALV 功能（ALV 服务）。

2）要分配一个单元格内字段的属性。

还可以对字段对象进行以下设置：

1）取得字段对象。

2）取得字段名称。

注：可以创建新的字段或删除现有的字段。

（1）取得字段对象

要为一个字段进行必要的设置，必须先得到该字段的实例。可以决定是否针对特定的字段对象（通过名称取得）或得到所有的字段对象（一个一个进行处理）。在这两种情况下，可以使用接口类 IF_SALV_WD_FIELD_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-6。

表 6-6

功　　能	方　　法
取得单个字段对象	GET_FIELD
取得所有字段对象	GET_FIELDS

（2）取得字段名称

取得当前字段实例的名称，可以使用类 CL_SALV_WD_FIELD，见表 6-7。

表 6-7

功　　能	方　　法
取得字段名称	GET_FIELDNAME

（3）创建或删除字段对象

调用 ALV 时，相应的字段对象或参照指定的 Context 节点属性自动创建。但是，在特殊情况下，程序员可能需要创建或删除字段对象。

当创建一个字段对象时，需要输入它的技术名称。该名称必须与应用程序的 Context 节点中的属性的名称一致。可使用接口类 IF_SALV_WD_FIELD_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）中的方法，见表 6-8。

表 6-8

功　　能	方　　法
创建字段对象	CREATE_FIELD
删除字段对象	DELETE_FIELD

6.2.3　列

根据如何取得的 ALV 配置模型，要么只有一个与 Context 节点属性相同名称的列对象，要么没有列对象。如果没有列对象，则必须创建显示 ALV 输出所需要的列对象。

对于列对象，可以进行以下设置：

- 取得某一列对象。
- 取得列对象的技术名称。
- 创建和删除某一列对象。
- 设置列标题。
- 更改某一列对象的位置。

（1）取得某一列对象

要为某一列进行必要的设置，首先必须获得该列的实例。可以根据名称针对特定的列对象进行设置或一次获得所有的列对象，然后一个一个处理。在这两种情况下，都可以使用接口类 IF_SALV_WD_COLUMN_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-9。

表 6-9

功　　能	方　　法
取得单独列对象	GET_COLUMN
取得所有列对象	GET_COLUMNS

（2）取得列对象的技术名称

使用的类 CL_SALV_WD_COLUMN 得到当前列的实例的名称，见表 6-10。

表 6-10

功　　能	方　　法
取得技术名称	GET_ID

（3）创建和删除某一列对象

如果取得 ALV 配置模型时，所指定的系统没有生成列对象，则需要创建应用程序进行 ALV 数据显示所需的列对象。

列对象存在的所有列显示在一个列集中。如果不希望某一列出现在用户的列集中，则必须删除相应的列对象。

注：当创建一个列对象时，需要输入它的技术名称。该名称必须与应用程序 Context 节点中的属性的名称一致。

要创建或删除列对象，可使用接口类 IF_SALV_WD_COLUMN_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-11。

表 6-11

功　能	方　法
创建一个列对象	CREATE_COLUMN
删除一个列对象	DELETE_COLUMN
删除所有列对象	DELETE_COLUMNS

6.2.3.1　列标题

默认情况下，ALV 输出中的每一列中有一列标题。不必因此必须首先创建相应的列对象（可以多个列用一个标题）。

可以更改列标题。更改时可以设置以下组件：文本、图形、工具提示。

可以为列标题进行如下设置：

- 创建、取得、删除列对象的列标题。
- 指定列标题的文字。
- 指定列标题中的图形路径。
- 在列标题中指定图形位置。
- 指定列标题文字的工具提示。

（1）关于标准列标题的信息

标准列标题包含文字内容。所使用的文字依赖于 Context 节点的属性。

1）如果属性没有 DDIC 引用，默认情况下，系统使用属性的技术名称作为列标题。

2）如果属性有一个 DDIC 引用，系统分配的列标题如下：

- 如果已定义 DDIC 数据元素，则系统采用 DDIC 数据元素的短文本字段标签作为列标题。
- 如果没有被定义字段标签，则系统采用 DDIC 数据元素的技术名称作为列标题。

（2）创建、取得、删除列对象的列标题

ALV 输出中的列的列标题是类 CL_SALV_WD_COLUMN_HEADER 的实例。列标题对象可以存在于每一列对象中。

要取得、创建或删除列标题对象，可以使用类 CL_SALV_WD_COLUMN 中的方法，见表 6-12。

表 6-12

功　能	方　法
取得列标题对象	GET_HEADER
创建列标题对象	CREATE_HEADER
删除列标题对象	DELETE_HEADER

（3）指定列标题的文字

指定要显示的列标题的文本，可有以下几种方法：

1）输入用户定义的文本。

2）输入一个单独的 DDIC 数据元素被用来作为列标题的字段标签。

对于和 DDIC 关系，首先以 DDIC 的数据元素作为切入点，返回列标题的文本。为此，需要使用方法 SET_PROP_DDIC_BINDING_ELEMENT，完成以下工作：

1）列标题或工具提示。DDIC 的文本说明是用来作为列标题的。

2）DDIC 数据元素的名称。默认情况下，使用 DDIC 数据元素的短文本。还可以使用 DDIC 数据元素的 heading 文本，medium 文本或长文本为此，需要使用方法 SET_PROP_DDIC_BINDING_FIELD。

注：如果输入用户定义的文本作为列标题，则必须使用该方法显式地关闭与 DDIC 的关系。

要指定列标题的文本，可以使用类 CL_SALV_WD_COLUMN_HEADER 中的方法，见表 6-13。

表 6-13

功　　能	方　　法
指定任何文字作为列标题	SET_TEXT
明确取消与 DDIC 的关系	SET_PROP_DDIC_BINDING_FIELD
取得列标题的文本	GET_TEXT
指定 DDIC 数据元素的字段标签用来作为列标题	SET_PROP_DDIC_BINDING_ELEMENT
取得 DDIC 数据元素的名称	GET_ PROP_DDIC_BINDING_ELEMENT
指定字段标签的类型，用来作为列标题	SET_ PROP_DDIC_BINDING_FIELD
取得字段标签的类型用作列标题	GET_ PROP_DDIC_BINDING_FIELD

（4）列标题换行

默认情况下，系统在列标题中不添加换行符。因此，列的宽度由列标题中的文本的长度决定。在列标题中，程序员可以指定换行符。如果列标题包含的字符允许换行（如空格或连字符）或者列不够宽，系统可以把文字拆成多行。要做到这一点，可以使用类 CL_SALV_WD_COLUMN_HEADER 的方法，见表 6-14。

表 6-14

功　　能	方　　法
设定换行	SET_HEADER_TEXT_WRAPPING
确定列标题中是否有换行设置	GET_HEADER_TEXT_WRAPPING

（5）指定列标题中的图形路径

可以在列标题中显示任何图形，只要该图形兼容 Web Dynpro 环境。要指定图形文件的路径或 ID，可以使用类 CL_SALV_WD_COLUMN_HEADER 方法，见表 6-15。

表 6-15

功　　能	方　　法
指定图形的路径或 ID	SET_IMAGE_SOURCE
取得图形的路径或 ID	GET_IMAGE_SOURCE

（6）在列标题中指定图形位置

在列标题中可以选择是否将图形显示在列标题的文本之前或之后。要做到这一点，可以使用类 CL_SALV_WD_COLUMN_HEADER 中的方法，见表 6-16。

表 6-16

功　　能	方　　法
指定在标题内图形的位置	SET_IMAGE_FIRST
取得在标题内图形的位置	GET_IMAGE_FIRST

（7）指定列标题文字的工具提示

当用户把光标移到列标题时，列标题显示工具提示。

指定要显示的提示文字有以下几种方式：

- 可以指定任何文字。
- 可以输入一个单独的 DDIC 数据元素的字段标签用来作为工具提示。

注：除了方法之外，设置标题工具提示与 DDIC 之间的关系一样，同样可以利用 DDIC 数据元素的短、中、长及标题标签文本作为工具提示文本。

要指定提示的文本，使用类 CL_SALV_WD_COLUMN_HEADER 中的方法，见表 6-17。

表 6-17

功　　能	方　　法
指定任何文字作为列标题工具提示文本	SET_TOOLTIP
明确取消与 DDIC 的关系	SET_PROP_DDIC_BINDING_FIELD
取得列标题工具提示的文本	GET_TOOLTIP
指定 DDIC 数据元素的字段标签用来作为列标题工具提示文本	SET_PROP_DDIC_BINDING_ELEMENT
取得 DDIC 数据元素的名称	GET_ PROP_DDIC_BINDING_ELEMENT
指定字段标签的类型，用来作为列标题工具提示文本	SET_ PROP_DDIC_BINDING_FIELD
取得字段标签的类型用作列标题工具提示文本	GET_ PROP_DDIC_BINDING_FIELD

6.2.3.2　列的位置

默认情况下，所有列的排序与应用程序的 Context 节点中的属性顺序相同。程序员可以通过以下方式更改列的顺序：通过改变位置号码、固定列。也可以指定是否允许用户来固定列。

（1）改变位置号码

系统自动分配给每一列一个位置编号。可以通过改变这个位置编号更改位置的列。这里的位置编号不是唯一的。

注：位置编号为 0 的列在最左边。这意味着，如果要把某一列放到左侧，则必须给一个

比其他列较高的位置编号。

也可以使用负数的位置编号。这意味着，给定某一列适当的位置，不改变所有其他列的0的位置编号。

要改变列的位置编号，可以使用类CL_SALV_WD_COLUMN中的方法，见表6-18。

表 6-18

功　能	方　法
设定位置编号	SET_POSITION
取得位置编号	GET_POSITION

（2）固定列

可以将重要列固定。这具有以下效果：

- 将列放在ALV输出的边缘。
- 滚动侧身的水平滚动条列可以不再被移动。

一列固定时，可以在ALV输出中指定该列是否是被固定到的左手侧或右手侧。

在这种方式中，最多可以创建以下3个区块的列：列固定到左，列是不固定的，列固定到右。

还可以更改一列的位置编号，以固定其位置。这使程序员可以根据自己的位置编号排列所有列块。

要固定某一列，可以使用类CL_SALV_WD_COLUMN中的方法，见表6-19。

表 6-19

功　能	方　法
固定某一列	SET_FIXED_POSITION
检查某一列是否被固定，并返回其固定位置	GET_FIXED_POSITION

（3）允许列固定

可以指定是否允许用户固定个别列。指定这些列分别固定到右边缘或左边缘。在ALV输出页面中单击“设置”按钮，在弹出对话框中，用户可以指定固定列的数量。

为了让列可以固定，可以使用接口类IF_SALV_WD_STD_FUNCTIONS（实现类CL_SALV_WD_TABLE）的方法，见表6-20。

表 6-20

功　能	方　法
允许列在ALV输出的左边缘固定	SET_FIXED_COLS_LEFT_ALLOWED
检查用户是否被允许在ALV输出的左边缘固定列	IS_FIXED_COLS_LEFT_ALLOWED
允许列在ALV输出的右边缘固定	SET_FIXED_COLS_RIGHT_ALLOWED
检查用户是否被允许在ALV输出的右边缘固定列	IS_FIXED_COLS_RIGHT_ALLOWED

6.2.4　页脚和滚动条

ALV输出的页脚中总是包含垂直滚动条（paginators），也可以包含水平（滚动条）（paginators）。

可以对页脚和滚动条进行以下设置：

- 显示或隐藏页脚。
- 显示水平滚动条。
- 水平或垂直滚动。

（1）显示或隐藏页脚

ALV 输出第一次显示一般有太多的条目。用户不使用滚动条就无法查看后面的条目；相反，必须在页脚使用的滚动条导航到所需的位置。

如果 ALV 输出包含大量的数据记录，则程序员必须确保为用户提供包含滚动条的页脚。

程序员还可以控制页脚中的滚动条什么时候可见，什么时候不可见。

1）不可见：即使项目被隐藏在看不见的区域，页脚也不显示。

2）始终可见：即使所有的条目在任何时间都是可见的，显示页脚。在这种情况下，滚动条被禁用。

3）仅在需要时可见。如果在 ALV 输出中有更多的行和列可以显示，则才显示页脚。

要显示或隐藏页脚，使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）中的方法，见表 6-21。

表 6-21

功　　能	方　　法
显示页脚	SET_FOOTER_VISIBLE
检查页脚是否显示	GET_FOOTER_VISIBLE

（2）显示水平滚动条

默认情况下，所有列标记为可见的显示方式。用户可以在浏览器窗体中使用水平滚动条浏览到所需的位置。

程序员可以定义滚动多少所有列才是可见的，从而确定 ALV 输出的宽度。用户需要使用滚动条顺序移动不可见列到可见区。

注：无论选择多少可见行，垂直滚动条总是可见的。

然而，水平滚动条在定义滚动列数后才可见。如果设置这个数字足够高使得所有列是可见的，则滚动条被关闭。

定义可见列的数量，从而确定是否要显示或隐藏滚动条，使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-22。

表 6-22

功　　能	方　　法
定义可滚动列数	SET_SCROLLABLE_COL_COUNT
取得可滚动列数	GET_SCROLLABLE_COL_COUNT

（3）水平或垂直滚动

用户需要滚动条按钮，以推动 ALV 输出中看不见的列或行到可见区域。用户也可以在 ALV 输出时，使用滚动条输入字段输入要被显示的行或列为第一个可见的行或列。

无论是否显示页脚，用户都可以使用 ALV 配置模型来定义显示的行或列。

注：要指定第一行，需使用该行的索引。指定第一列时，需使用该列的技术名称。

在 ALV 输出中要滚动到所需的列或行，使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_CLASS）的方法，见表 6-23。

表 6-23

功　　能	方　　法
指定要被显示的行为第一行	SET_FIRST_VISIBLE_ROW
取得第一行	GET_FIRST_VISIBLE_ROW
指定要被显示的列为第一列	SET_FIRST_VISIBLE_SCROLL_COL
取得第一列	GET_FIRST_VISIBLE_SCROLL_COL

6.2.5　页眉与页脚区域

用户可以在 ALV 输出的上方和下方使用各种元素设计页眉与页脚。可以使用设计对象进行相关操作。

程序员可以使用许多不同的类型的元素，以形成一个设计对象，然后显示在想要的位置。

注：所有设计对象的类和方法在系统内的SALV_FORM_ELEM包中。

（1）元素类型和布局形式

可以使用下列类型的元素来设计对象：

- Header 元素（头信息）。
- 带或不带标签的文本元素（文本标签）。
- 动作信息。

注：这些元素只在外观上有所不同。各类型元素没有相关功能连接。

这些元素可以排列在设计对象中。可以选择的布局表单有以下 3 种：

- 单元素：生成一个元素，并将其显示在所需的位置。
- Row-type 布局：排列尽可能多的元素，将其一前一后排成一排。
- Table-type 布局：安排尽可能多的元素，将其排列成行和列。

注：可以将二者结合起来布局表单，如可以将行插入到一个表中。

（2）Context 节点 TOP_OF_LIST 和 END_OF_LIST

ALV 组件提供了两个 Context 节点 TOP_OF_LIST 和 END_OF_LIST。每个节点包含一个 CONTENT 属性。这些 Context 节点用于 ALV 输出中保存页眉和页脚区域的设计对象的数据。

可以定义 Context 映射到应用程序一个相同的名称 Context 节点上，并将该 Context 设置到设计对象的属性。

（3）设置设计对象（Design Object）

设置设计对象的页眉和页脚区域的步骤如下：

1）映射 ALV 组件的 Context 节点 TOP_OF_LIST 和 END_OF_LIST 到应用程序 Context 中。

2）创建一个设计对象。要做到这一点，可以使用 row-type 或 table-type 布局。将所需的元素插入到这个布局中。

3）如有需要，将元素进行各种设置（设计对象设置）。

4）将设计对象设置为 Context 节点的 CONTENT 属性。

（4）设计对象的显示和隐藏

默认情况下，页眉区设计对象和页脚区的设计对象都显示出来。

隐藏和显示这两种设计对象，可以使用接口 IF_SALV_WD_TABLE_SETTINGS（实现类的 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-24。

表 6-24

功　能	方　法
显示或隐藏的页眉区域设计对象	SET_TOP_OF_LIST_VISIBLE
检查页眉区域设计对象是否显示或隐藏	GET_TOP_OF_LIST_VISIBLE
显示或隐藏的页脚区域设计对象	SET_END_OF_LIST_VISIBLE
检查页脚区域设计对象是否显示或隐藏	GET_END_OF_LIST_VISIBLE

6.2.5.1　创建设计对象和元素

（1）创建一个设计对象

首先，确定设计对象的基本布局。在显示时，可以决定在一行里显示一个单一的元素或者一排的元素序列，或将多个元素按照某种顺序设置到多个行和列中。使用表 6-25 中的类用于此目的。

表 6-25

布　局	类
Row-type 布局	CL_SALV_FORM_LAYOUT_FLOW
Table-type 布局	CL_SALV_FORM_LAYOUT_GRID

当创建一个标签类型的元素时，必须指定相应的文本元素。这意味着程序员不能使用这个元素作为设计对象中的单个元素。

（2）创建单个元素

已经创建了一个布局设计对象后，可以使用表 6-26 中的方法来创建各种类型的元素：这些方法可以在类 CL_SALV_FORM_LAYOUT_FLOW 和 CL_SALV_FORM_LAYOUT_GRID 中找到。

表 6-26

功　能	方　法
创建 Row-type 类型布局（可用于嵌套）	CREATE_FLOW
创建 Table-type 类型布局（可用于嵌套）	CREATE_GRID
创建文本元素	CREATE_TEXT
创建特定的文本元素的标签	CREATE_LABEL
创建页眉元素	CREATE_HEADER_INFORMATION
创建动作信息	CREATE_ACTION_INFORMATION

可以使用表 6-27 中的类创建对象的元素。

表 6-27

元 素 类 型	类
Row-type 类型布局（可用于嵌套）	CL_SALV_FORM_LAYOUT_FLOW
Table-type 类型布局（可用于嵌套）	CL_SALV_FORM_LAYOUT_GRID
文本元素	CL_SALV_FORM_TEXT
标签元素	CL_SALV_FORM_LABEL
标题信息	CL_SALV_FORM_HEADER_INFO
动作信息	CL_SALV_FORM_ACTION_INFO

注：行类型的布局的创建不需要参数。表型布局创建时需指定行和列的参数。当创建一个元素类型标签时，必须指定相应的文本元素（R_LABEL_FOR），无论所处的布局类型。

6.2.5.2 设定设计对象

程序员可以对设计对象作相关设定，并可以为每个设计对象包含的元素作相关设定，在 table-type 布局的情况下，也可以为单独的列作相关设定。

（1）设计对象设置

程序员可以对设计对象元素进行以下设置，见表 6-28（布局为 Flow 时）和表 6-29（布局为 Grid 时）。

表 6-28

功 能	方 法
设定布局内元素	SET_ELEMENT
取得 Row-type 布局中的元素数	GET_ELEMENT_COUNT
设定工具提示的设计目标	SET_TOOLTIP
取得工具提示为设计对象	GET_TOOLTIP

表 6-29

功 能	方 法
设定布局内元素	SET_ELEMENT
附加的空行（FLOW 对象）	ADD_ROW
取得的行数	GET_ROW_COUNT
显示列和行之间的网格	SET_GRID_LINES
设定工具提示的设计目标	SET_TOOLTIP
取得工具提示为设计对象	GET_TOOLTIP

（2）设定元素

可为不同类型的元素设定相似的功能，见表 6-30。

表 6-30

功　能	方　法	类
设定元素的文本	SET_TEXT	CL_SALV_FORM_TEXT CL_SALV_FORM_LABEL CL_SALV_FORM_HEADER_INFO CL_SALV_FORM_ACTION_INFO
取得元素的文本	GET_TEXT	
设定元素的工具提示	SET_TOOLTIP	
取得元素的工具提示	GET_TOOLTIP	
设置相应的文本元素	SET_LABEL_FOR	CL_SALV_FORM_LABEL
取得相应的文本元素	GET_LABEL_FOR	

（3）设定列的宽度及对齐

一列在 Table-type 布局中的设计对象是类 CL_SALV_FORM_GRID_COLUMN 的一个对象。可以使用类 CL_SALV_FORM_LAYOUT_GRID 一次创建多个列的对象，具体方法见表 6-31。

表 6-31

功　能	方　法
取得列	GET_COLUMN
在表布局中创建列	SET_COLUMN_COUNT
取得表布局中的列数	GET_COLUMN_COUNT

可以使用列对象来定义列的宽度和对齐列中的元素，具体方法见表 6-32。

表 6-32

功　能	方　法
设定列宽	SET_WIDTH
取得列宽	GET_WIDTH
设定元素水平对齐方式	SET_H_ALIGN
取得元素水平对齐方式	GET_H_ALIGN

（4）单一元素的宽度和对齐方式设定

可以定义个别元素的宽度和对齐方式。由于技术原因，必须确定该元素是否在 Row-type 布局或 Table-type 布局。

- 对于 Row-type 布局，使用类 CL_SALV_FORM_LAYOUT_DATA_FLOW。
- 对于 Table-type 布局，使用类 CL_SALV_FORM_LAYOUT_DATA_GRID。

要更改布局数据，必须首先确定布局类型。取得元素布局的方法见表 6-33。

表 6-33

功　能	方　法
获取元素的对齐和宽度的布局数据	GET_LAYOUT_DATA

要定义元素的宽度和对齐方式，现在使用的相应类的方法，见表 6-34。

表 6-34

功　　能	方　　法
设定宽度	SET_WIDTH
取得宽度	GET_WIDTH
设定水平对齐	SET_H_ALIGN
取得水平对齐	GET_H_ALIGN

6.2.5.3　设计对象实例

（1）table-type 布局 ALV 输出的页眉

下面的示例演示如何在 Table-type 布局的 ALV 输出的页眉中插入一个设计对象：

```
...
*... TOP_OF_LIST
data:
lr_node type ref to if_wd_context_node,
lr_grid type ref to cl_salv_form_layout_grid,
lr_text type ref to cl_salv_form_text,
lr_label type ref to cl_salv_form_label.

create object lr_grid.
lr_text = lr_grid->create_text(
text = '1.2 TEXT'
row = 1
column = 2 ).
lr_label = lr_grid->create_label(
text = '1.1 LABEL'
row = 1
column = 1
r_label_for = lr_text ).
lr_text = lr_grid->create_text(
text = '2.2 TEXT'
row = 2
column = 2 ).
lr_label = lr_grid->create_label(
text = '2.1 LABEL'
row = 2
column = 1
r_label_for = lr_text ).
lr_node = wd_context->get_child_node( name = 'TOP_OF_LIST' ).

call method lr_node->set_attribute
exporting
value = lr_grid
name = 'CONTENT'.
...
```

（2）Row-Type 布局 ALV 输出的页脚

下面的例子演示了如何在 ALV 输出的页脚中显示一个文本类型的元素：

```
...
*... END_OF_LIST
data:
lr_node type ref to if_wd_context_node,
lr_flow type ref to cl_salv_form_layout_flow,
lr_text type ref to cl_salv_form_text.
create object lr_flow.
lr_text = lr_flow->create_text(
text = 'MyText').
lr_node = wd_context->get_child_node( name = 'END_OF_LIST' ).
call method lr_node->set_attribute
exporting
value = lr_flow
name = 'CONTENT'.
...
```

6.2.6 设定对话框

一些 UI 元素的 ALV 标准功能都位于一个特殊的领域：“设置”对话框。

默认情况下，“设置”对话框显示为一个 ALV 输出上方的 ALV 组件接口的接口视图服务。程序员可以为应用程序指定是否以对话框形式显示，也就是说，是否在 ALV 输出上方作为一个弹出窗口。

ALV 输出时对话框被隐藏。用户可以把“设置”对话框放置在工具栏上。

（1）对话框的按钮

用户在该对话框中进行设置，可以用以下两种方法应用到 ALV 输出，如图 6-2 所示。



图 6-2

1）使用“应用”按钮：适用于 ALV 输出的设置。该对话框保持打开状态，用户可以进行进一步的设定。

2）使用“确定”按钮：适用于 ALV 输出的设置。对话框关闭。

当用户单击“取消”对话框时，关闭对话框，ALV 输出保持不变。

（2）显示一个弹出的对话框

作为一个单独的模式对话框显示“设置”对话框，可以使用接口类 IF_SALV_WD_STD_FUNCTIONS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-35。

表 6-35

功　　能	方　　法
显示一个弹出式对话框	SET_DIALOG_SETTINGS_AS_POPUP
检查是否显示为一个弹出的对话框	IS_DIALOG_SETTINGS_AS_POPUP

6.3 ALV 输出的外观

程序员可以用各种方式更改 ALV 输出的外观。可以指定以下内容：

- ALV 输出，列和单元格的大小。
- 个别区域的可见性。
- ALV 输出，列和单元格的颜色。
- 文本属性。
- 列和行之间的网格。
- 树状层次结构表。
- 表数据作为业务图形输出。

6.3.1 将属性分配给列和单元格

可以通过以下两种分配方式影响 ALV 列输出外观的大部分的属性：

1）将属性分配到整个列。列中的每个单元格都具有这个属性。

2）在每一个单元格中分配不同的字段，当该单元格出现在当前列中时将指定的这个字段分配给当前列。

这使程序员可以指定所需的属性到单独的单元格。

程序员还可以指定下面的属性：

1）列的背景颜色（SET_CELL_DESIGN 或 SET_CELL_DESIGN_FIELDNAME）。

2）单元格的变式（SET_SELECTED_CELL_VARIANT 或 SET_SEL_CELL_VARIANT_FIELDNAME）。

影响一个单元格外观的一些属性与单元格中的单元格编辑器相关联。这意味着单元格编辑器的属性确定如何显示一列：对于编辑器的所有实例，这些属性可以是有效的，也可以被一个字段的信息覆盖。

例子：

在一列中，程序员使用 BUTTON 单元编辑器，每个单元被显示为一个按钮，但是要在此列中隐藏一些单元格。

由于 visibility 属性是基于按钮的，而不是基于列的，因此必须使用一种变通方法隐藏单元格。

1）参照类 CL_SALV_WD_UIE_BUTTON 创建一个对象（如 lr_button）。

2）定义一个 Context 字段（如 BUTTON_VIS），该字段的信息包含每个单元格的可见性。分配这一字段到按钮 lr_button：

```
lr_button-> set_visible_fieldname ('BUTTON_VIS').
```

3）指定按钮的单元编辑器的列（如 lr_column）：

```
lr_column - > set_cell_editor (lr_button).
```

这意味着，BUTTON_VIS 字段确定在该列的单元格 lr_column 按钮是可见的。

示例：

```
LOOP AT EL_ELEMENTS INTO EL_ELEMENT.
  EL_ELEMENT->GET_ATTRIBUTE( EXPORTING NAME  = 'CARRID'
                  IMPORTING VALUE = LWA_SFLIGHT-CARRID ).

  CLEAR LWA_SFLIGHT-BUTTON_VIS.
  IF LWA_SFLIGHT-CARRID = 'LH'.
   LWA_SFLIGHT-BUTTON_VIS = '02'."cl_wd_table_column=>e_cell_design.
  ENDIF.
  APPEND LWA_SFLIGHT TO T_SFLIGHT.
ENDLOOP.
LR_BUTTON->SET_VISIBLE_FIELDNAME( 'BUTTON_VIS' ).
```

使用单元格编辑器指定属性的其他例子：

- 用于 TEXT_VIEW 字体类型的设置（SET_DESIGN 或 SET_DESIGN_FIELDNAME）。
- 一个选定切换按钮图形的设置（SET_CHECKED_IMAGE_SOURCE 或 SET_CHECKED_IMG_SRC_FIELDNAME）。
- 用于 ProgressIndicator 的尺寸进度条的设定（SET_PERCENT_VALUE 或 SET_PERCENT_VALUE_FIELDNAME）。

在该过程中的选项依赖于单元格编辑器的 UI 元素处理。

6.3.2 ALV 输出、列和单元格的大小

默认情况下，ALV 输出，列和行的大小取决于其内容。这意味着：

- 每一列的宽度至少为最宽的单元格的宽度。
- ALV 输出宽度是所有列宽度的和。
- 每一行的高度至少为最高的单元格的高度。

程序员通过指定所需的宽度可以轻松地增加 ALV 输出和列的宽度。然而将宽度减小是不容易的。首先，需要冻结 ALV 输出的布局，这会分配所有列相同的宽度；然后，指定所需的每一列的宽度。可以进行以下设置，限定各个领域的大小：改变宽度、改变高度、冻结表格布局。

注：只能控制使用它们的内容（如使用的图形的尺寸或多行文本）的行的高度。

（1）改变宽度

要改变一个区域的宽度，使用该区域类的方法，见表 6-36。

表 6-36

功　　能	类	方　　法
指定 ALV 输出的宽度	IF_SALV_WD_TABLE_SETTINGS	SET_WIDTH
指定列的宽度	CL_SALV_WD_COLUMN	
指定业务图形的宽度	IF_SALV_WD_GRAPHIC_SETTINGS	
指定单元格的宽度（只有在下面的单元格编辑器）		SET_WIDTH 或 SET_WIDTH_FIELDNAME
按钮（Button）	CL_SALV_WD_UIE_BUTTON	
下拉列表（Dropdown list box）	CL_SALV_WD_UIE_DROPDOWN_BY_KEY	
图形（Image）	CL_SALV_WD_UIE_IMAGE	
输入框（InputField）	CL_SALV_WD_UIE_INPUT_FIELD	
进度指示器（Progress Indicator）	CL_SALV_WD_UIE_PROGR_INDICATOR	
切换按钮（Toggle Button）	CL_SALV_WD_UIE_TOGGLE_BUTTON	
值比较（Value Comparison）	CL_SALV_WD_UIE_VALUE_CMP	
取得 ALV 输出的宽度	IF_SALV_WD_TABLE_SETTINGS	GET_WIDTH
取得列宽	CL_SALV_WD_COLUMN	
取得商业图表的宽度	IF_SALV_WD_GRAPHIC_SETTINGS	
取得单元格的宽度（只有在上面的单元格编辑器）		GET_WIDTH 或 GET_WIDTH_FIELDNAME

（2）改变高度

使用在一个时间段内显示的行的数量来指定 ALV 输出的高度。

注：如果数据记录数小于可见行数，则空行会自动添加到 ALV 输出底部。可以指定没有空行可以追加，则 ALV 输出的高度会根据本身的数据记录数加以调整。

要做到这一点，可以使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-37。

表 6-37

功　　能	方　　法
指定可见行数	SET_VISIBLE_ROW_COUNT
取得可见行数	GET_VISIBLE_ROW_COUNT
追加空行	SET_DISPLAY_EMPTY_ROWS
检查是否被追加空行	GET_DISPLAY_EMPTY_ROWS

注：要显示所有内部数据表中的数据记录，将可见行数设置为-1。

（3）冻结表布局

冻结 ALV 输出的布局可以用来使列的宽度窄于它们内容的长度。

要做到这一点，可以使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-38。

表 6-38

功　　能	方　　法
冻结表格布局	SET_FIXED_TABLE_LAYOUT
检查表格布局是否被冻结	GET_FIXED_TABLE_LAYOUT

6.3.3　个别区域的可见性

可以显示或隐藏 ALV 输出的不同区域。

注：如果一个区域是在一个不可见的区域内，则该区域也是不可见的。

可以显示和隐藏以下几个方面的整个区域或某个区域内的个别区域，见表 6-39。

- 整个 ALV 输出。
- 单个列。
- 作为单元格编辑器和单元格变式的每个 UI 元素。
- 整个工具栏及所有功能。
- 在工具栏上个别标准 ALV 功能的 UI 元素。
- ALV 输出中页眉区域的设计对象。
- ALV 输出中页脚区域的设计对象。

注：若要隐藏其中的对象，如 ALV 输出的页眉或列标题，则必须删除相关的对象。

表 6-39

<table>
<tr><th>区　　域</th><th>类</th><th>方　　法</th></tr>
<tr><td>整个 ALV 输出</td><td>IF_SALV_WD_TABLE_SETTINGS</td><td rowspan="4">SET_VISIBLE
GET_VISIBLE</td></tr>
<tr><td>单个列</td><td>CL_SALV_WD_COLUMN</td></tr>
<tr><td>作为单元格编辑器和单元格变式的每个 UI 元素</td><td>CL_SALV_WD_UIE(所有的 UI 元素的基类)</td></tr>
<tr><td>整个工具栏及所有功能</td><td>IF_SALV_WD_FUNCTION_SETTINGS</td></tr>
<tr><td>ALV 输出中页眉区域的设计对象</td><td>IF_SALV_WD_TABLE_SETTINGS</td><td>SET_TOP_OF_LIST_VISIBLE</td></tr>
<tr><td>ALV 输出中页脚区域的设计对象</td><td>IF_SALV_WD_TABLE_SETTINGS</td><td>SET_END_OF_LIST_VISIBLE</td></tr>
<tr><td>在工具栏上个别标准 ALV 功能的 UI 元素</td><td>IF_SALV_WD_STD_FUNCTIONS</td><td>SET_<standard function>_ALLOWED
IS_<standard function>_ALLOWED</td></tr>
</table>

6.3.4　ALV 输出、列和单元格的颜色

可以指定各个领域的 ALV 输出的颜色。有以下选项可供选择：

1）ALV 输出。可以选择下列变量：

● 标准颜色分配。所有的行和列具有相同的颜色。
● 交替。ALV 输出行在明与暗之间交替。
● 透明。背景是透明的。各个单元格之间没有显示网格线。

2）列。可以为一列更改背景颜色。可以使用预先配置（pre-configured）的语义颜色。

3）单元格变式。可以指定单元格变式的背景颜色。可以使用预先配置（pre-configured）的语义颜色。

注：如果写保护（write-protection）被激活，则用户可以使用这些选项。

4）单元格。可以改变单元格编辑器 TEXT_VIEW 的颜色，也是唯一可以改变颜色的 UI 元素。可以选择使用背景和文字颜色的组合。

5）工具栏。可以选择下列变量：

● 标准颜色分配。
● 彩色。
● 透明。

改变颜色的方法见表 6-40。

表 6-40

区　　域	类	方　　法
ALV 输出	IF_SALV_WD_TABLE_SETTINGS	SET_DESIGN GET_DESIGN
列	CL_SALV_WD_COLUMN	SET_CELL_DESIGN GET_CELL_DESIGN
单元格变式	CL_SALV_WD_CV_STANDARD	SET_CELL_DESIGN 或 SET_CELL_DESIGN_FIELDNAME
		GET_CELL_DESIGN 或 GET_CELL_DESIGN_FIELDNAME
单元格	CL_SALV_WD_UIE_TEXT_VIEW	SET_SEMANTIC_COLOR 或 SET_SEMANTIC_COLOR_FIELDNAME
		GET_SEMANTIC_COLOR 或 GET_SEMANTIC_COLOR_FIELDNAME
工具栏	IF_SALV_WD_ FUNCTION_SETTINGS	SET_DESIGN GET_DESIGN

6.3.5　文本属性

可以在 ALV 输出中单个区域内指定如下文本的显示方式，见表 6-41。

1）在列的水平对齐方式。

2）一个单元格变式的水平对齐方式。

3）在单元格中的字体大小和风格。

该功能只适用于单元格编辑器 TEXT_VIEW。在这里，可以选择使用预先设定的格式。

● 在单元格中换行。

该功能仅适用于细胞编辑器 TEXT_VIEW、LINK_TO_ACTION 和 LINK_TO_URL。如果文本包含字符，则允许一个换行符（如空格和连字符），文本可以显示为多行。

● 一个单元格内字体的颜色。
● 页眉或页脚区域中设计对象的对齐、字体大小、字体样式。

表 6-41

功　能	类	方　法
列对齐	CL_SALV_WD_COLUMN	SET_H_ALIGN GET_H_ALIGN
单元格变式对齐	CL_SALV_WD_CV_STANDARD	SET_H_ALIGN GET_H_ALIGN
单元格字体大小和风格	CL_SALV_WD_UIE_TEXT_VIEW	SET_DESIGN 或 SET_DESIGN_FIELDNAME
		GET_DESIGN 或 GET_DESIGN_FIELDNAME
单元格中换行	CL_SALV_WD_UIE_TEXT_VIEW CL_SALV_WD_UIE_LINK_TO_ACTION CL_SALV_WD_UIE_LINK_TO_URL	SET_WRAPPING 或 SET_WRAPPING_FIELDNAME
		GET_WRAPPING 或 GET_WRAPPING_FIELDNAME

6.3.6　列和行之间的网格

在 ALV 输出中可以定义是否要显示或隐藏列和行之间的网格。可以选择下列变量：

- 仅显示列之间的网格。
- 仅显示行之间的网格。
- 显示行列之间的网格。
- 不显示网格。

可以使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-42。

表 6-42

功　能	方　法
显示或隐藏 Lines	SET_GRID_MODE
检查 Lines 是否显示或隐藏	GET_GRID_MODE

6.3.7　树状层次结构表

ALV 输出可以作为一个层次显示，其步骤如下：

- 指定层次结构中的列。可以指定一个或多个列的层次结构列。
- 指定显示类型。可以定义 ALV 输出的层次结构。

也可以指定所有的层级为可扩展的。默认情况下，当 ALV 输出显示时只有行的层次上是可见的。

（1）层次结构

作为层次结构显示 ALV 输出中列上具有以下的效果：

1）ALV 输出会自动根据所有层级列进行排序。

2）默认情况下，排序顺序（层级的序列）由列对象序列确定。程序员可以用以下不同的方式更改层级的顺序：

- 更改（层次）列的位置。
- 更改用于结构列顺序排序字段的层次。

3）定义的层次结构列的列不会以通常的形式显示。相反，所有的层次结构的所有列的值

一起被显示在第一列中。该值根据它所属的层级缩进。

4）一个层次结构的第一列中的值有一个小的箭头图标。用户可以利用这个箭头图标显示或隐藏所有较低级别的数据记录。

可以指定列中的数据中最后的层次结构不作为一个带有一个箭头图标节点显示，可用一个点作为叶子显示。

5）默认情况下，只有较高层次级别的行显示。较低层级不会被扩展，因此不显示。

6）第一列的层次结构不是一个类 CL_SALV_WD_COLUMN 的对象实例。

7）列的列标题、每个层次结构列的列标题被列在一起。

8）层次列或它们的值（因为本身是不显示的列）是不能隐藏的。

注：页面“设置”对话框的“Column Selection”选项卡中会显示绿色的层次结构中的列。用户不能将它们设为隐藏列。

（2）指定层次结构中的列

要定义层次结构中的列可使用接口类 IF_SALV_WD_COLUMN_HIERARCHY（实现类 CL_SALV_WD_COLUMN）的方法，见表 6-43。

表 6-43

功　能	方　法
指定层次结构中的列	SET_HIERARCHY_COLUMN
检查列是否是一个层次结构列	IS_HIERARCHIY_COLUMN

（3）指定显示类型

定义 ALV 输出为一个层次，从而确定显示的类型，可以使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-44。

表 6-44

功　能	方　法
指定显示类型	SET_DISPLAY_TYPE
取得显示类型	GET_DISPLAY_TYPE

（4）扩展较低层级

对于层次结构显示，ALV 输出第一次显示时加载的所有数据。大量的数据可能会导致加载时间长。默认情况下，只有行的层次被显示出来。较低层级不会被扩展，因此不显示。

程序员也可以指定扩大所有的层级。要做到这一点，可以使用接口类 IF_SALV_WD_TABLE_HIERARCHY（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-45。

表 6-45

功　能	方　法
展开所有的层级	SET_EXPANDED
检查是否所有的层级将扩大	IS_EXPANDED

（5）显示层次结构列数据为叶（Leaf）

要在第一列的层次结构中显示最后的层次结构中的数据列作为叶，使用接口类 IF_SALV_WD_TABLE_HIERARCHY（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-46。

表 6-46

功　　能	方　　法
显示数据的最后层次结构列作为叶	SET_LAST_HIER_COLUMN_AS_LEAF
检查最后的层次结构的数据列是否被显示为一个叶	IS_LAST_HIER_COLUMN_AS_LEAF

6.3.8　表数据作为业务图形输出

ALV 可以作为业务图形显示 ALV 输出的数据。用于业务图形时可以进行以下设置：

- 显示业务图形。
- 指定数据使用。
- 更改业务图形的外观。
- 允许或禁止业务图表显示。

注：如果允许用户显示数据的业务图形，则可以从一系列的图表类型中选择。大多数用户选择立式条形图或条形图来显示 ALV 输出的数据。

（1）显示业务图形

可以指定当 ALV 输出时是否显示业务图形。可以选择下列选项：

- 只显示为表。
- 显示表和业务图形。
- 仅显示业务图形。

要做到这一点，可使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-47。

表 6-47

功　　能	方　　法
指定显示的类型	SET_DISPLAY_AS
取得显示的类型	GET_DISPLAY_AS

（2）指定数据使用

业务图形显示的数据主要取决于是否以及在哪些列进行计算：

1）如果不进行任何计算：

- 第一列的字母数字类型的数据用于包含图表特性的坐标轴。立式条形图通常是 X 轴，条形图通常是 Y 轴。
- 第一列的数值类型数据用于包含关键数据的坐标轴。立式条形图通常是 Y 轴，条形图通常是 X 轴。

2）如果进行计算：

- 以字母数字的类型数据的第一列用于包含图表的特性的坐标轴。
- 在 X 轴上显示的涉及计算的每一列，用于包含数字关键数据的坐标轴。

3）如果生成的中间结果：

- 这是以每一列为一个标准的中间结果显示的轴，它包含图表的特性。如果指定一个以上的标准（多个列），则标准的组合被显示。
- 坐标轴上每列涉及计算的，包含数字关键数据的图表被显示。

可以通过移动相应列到 ALV 输出的左边指定哪些数据将要被显示在业务图形中，或通过列的计算，但不影响业务图形的内容。

（3）改变业务图形的外观

可以指定一个业务图形的外观：

- 指定业务图形的维度。
- 指定业务图形的类型。
- 指定业务图形的大小。

1）维度。

可以为业务图形设置以下维度：

- 2D，如图 6-3 所示。

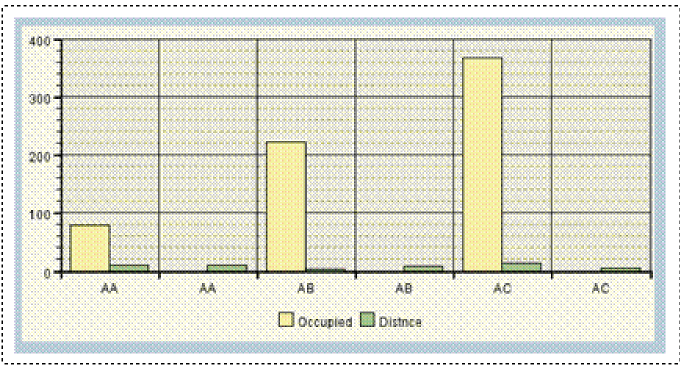


图 6-3

- 伪 3D，如图 6-4 所示。

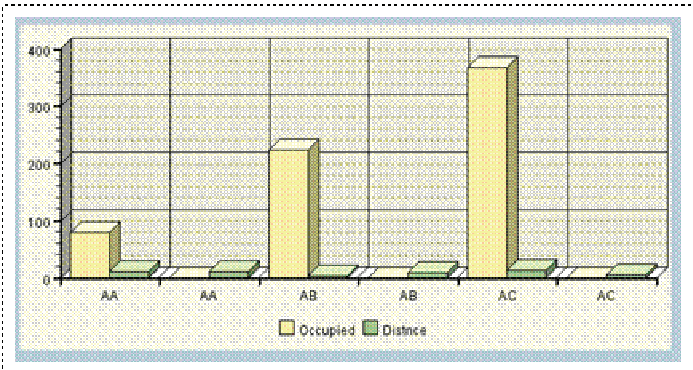


图 6-4

● 3D，如图 6-5 所示。

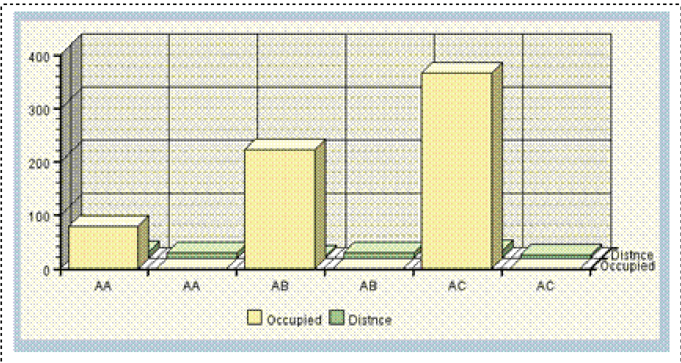


图 6-5

2）图表类型。

可以从以下的图表类型中进行选择：

- 面积图（Area chart）。
- 条形图（Bar chart）。
- 立式条形图（Vertical bar chart）。
- 圆环图（Doughnut chart）。
- 折线图（Line chart）。
- 饼图（Pie chart）。
- 分饼图（Split pie chart）。
- 堆积面积图（Stacked area chart）。
- 堆积条形图（Stacked bar chart）。
- 堆积立式条形图（Stacked vertical bar chart）。
- 堆积折线图（Stacked line chart）。

要改变业务图形的外观，使用接口类 IF_SALV_WD_GRAPHIC_SETTINGS（实施类 CL_SALV_WD_COLUMN）的方法，见表 6-48。

表 6-48

功　　能	方　　法
指定的商业图形的尺寸	SET_DIMENSION
获取业务的尺寸图形	GET_DIMENSION
指定高度的商业图表	SET_HEIGHT
获得高度的商业图表	GET_HEIGHT
指定图表类型	SET_TYPE
获得图表类型	GET_TYPE
指定业务宽度的图形	SET_WIDTH
获得宽度的商业图表	GET_WIDTH

（4）允许或不允许作为业务图形显示

默认情况下，允许用户在数据显示 ALV 输出业务图形。在工具栏的下拉列表框中，用户

可以从以下选项中进行选择：

- 只显示为表。
- 显示表和业务图形。
- 只显示业务图形。

可以禁止用户显示数据的业务图形。要做到这一点，可以使用接口类 IF_SALV_WD_STD_FUNCTIONS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法见表 6-49。

表 6-49

功　　能	方　　法
允许显示图形业务	SET_DISPLAY_AS_ALLOWED
检查是否允许商业图表显示	IS_DISPLAY_AS_ALLOWED

6.3.9　空输出的替换文本

如果没有可用的数据，则可以在 ALV 输出的区域显示指定的文本。

要为空 ALV 输出显示替换文本时使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-50。

表 6-50

功　　能	方　　法
指定替代文本	SET_EMPTY_TABLE_TEXT
替代文本	GET_EMPTY_TABLE_TEXT

6.4　预定义标准 ALV 功能

程序员可以在应用程序中预定义 ALV 在屏幕上的输出布局，当用户调用应用程序时该布局即被输出。该操作涉及数据本身及设计方面，因此程序员可以编辑该表中的数据以定义显示方式。相应功能如下：

- 排序。
- 过滤数据。
- 计算。
- 视图。
- 导出为外部格式数据。

6.4.1　排序

程序员可以根据特定的规则进行排序，更改数据记录的顺序，进行 ALV 输出。可以指定哪些字段的值按字母顺序或数字顺序（按大小）进行排序，从而确定序列中的所有行。

开发者可以进行以下排序设置：

- 创建、获取、删除排序设置（排序条件）。

- 设置排序方向和排序顺序。
- 按照相同的值分组。
- 强制某个字段的排序。
- 按照选中某一列启用排序。
- 根据另一个字段的值对一个字段结合排序。
- 隐藏和显示排序标签页。

注：程序员可以使用数字列的值进行合计。默认情况下，所有列的值均可用于此功能。除了合计之外，如果需要中间小计结果，则需要对ALV输出进行排序。如果要显示小计结果，则ALV可以按照列的默认排序情况自动生成小计结果。

（1）创建、获取、删除排序设置

排序是 ALV 输出中一个字段的属性。创建、获取或删除字段排序条件，可以使用接口类 IF_SALV_WD_SORT（实施类 CL_SALV_WD_FIELD）中的方法，见表 6-51。

表 6-51

功　　能	方　　法
取得排序条件	GET_SORT_RULE
创建排序条件	CREATE_SORT_RULE
删除排序条件	DELETE_SORT_RULE

字段的排序条件对象是类CL_SALV_WD_SORT_RULE的对象。

注：如果指定的ALV输出是一个新的结构，则新的数据内表的所有字段的排序条件会自动删除。

（2）设置排序方向和排序顺序

对于每个单独的排序条件，可以指定是否要排序及按字段值升序（A、B、C）或降序（C、B、A）排序（排序方向）。

如果通过改变由多个字段排序更改ALV输出，则输出结果依赖于排序序列中的哪些字段。默认情况下，该字段在序列中被排序以生成排序条件。程序员可以更改此序列。要做到这一点，可以为排序顺序的字段指定一个位置编号（或排序条件）。

若要更改排序条件的排序方向或排序顺序，则可以使用类 CL_SALV_WD_SORT_RULE 中的方法，见表 6-52。

表 6-52

功　　能	方　　法
指定排序方向	SET_SORT_ORDER
取得排序方向	GET_SORT_ORDER
设置排序顺序领域内的位置	SET_SORT_POSITION
取得排序顺序领域内的位置	GET_SORT_POSITION

（3）按照相同的值分组

默认情况下，在一个排序字段的排序的序列中，ALV 输出的值是相同的。ALV 自动将值

进行分组。可以指定一个值出现在每一行，即使这个值不会改变。要做到这一点，使用接口类 IF_SALV_WD_SORT（实现类 CL_SALV_WD_FIELD）的方法，见表 6-53。

表 6-53

功　　能	方　　法
创建/删除分组	SET_GROUPING_ALLOWED
检查是否对值进行分组	IS_GROUPING_ALLOWED

（4）强制某个字段的排序

可以明确地禁止一个字段进行排序，这具有以下效果：

- 相关领域不再显示的“Settings”对话框中的“排序”选项卡页面。
- 如果想通过用户单击列标题进行排序，则该功能不可用，禁止排序的列的箭头图标被隐藏。
- 如果在应用程序中为这一字段定义了一个排序条件，则这种状况不影响 ALV 输出。

要禁止某个字段的排序，使用接口类 IF_SALV_WD_SORT（实现类 CL_SALV_WD_FIELD）的方法，见表 6-54。

表 6-54

功　　能	方　　法
禁止排序	SET_SORT_ALLOWED
检查是否允许排序	IS_SORT_ALLOWED

（5）按照选中某一列启用排序

ALV 输出中的列标题中可以显示小箭头图标。用户可以使用这些箭头图标将列按升序或降序进行排序。

也可以指定按一列排序，及使用〈Crtl〉键添加多个列进行排序。

为了使用户能够通过单击列标题进行排序，可以使用接口的类 IF_SALV_WD_STD_FUNCTIONS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-55。

表 6-55

功　　能	方　　法
显示列标题中的箭头图标	SET_SORT_HEADERCLICK_ALLOWED
检查是否箭头图标显示在列标题中	IS_SORT_HEADERCLICK_ALLOWED

为了使用户能够通过单击列标题的多个列进行排序，可以使用接口类 F_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-56。

表 6-56

功　　能	方　　法
启用多个列的排序	SET_MULTI_COLUMN_SORT
检查是否按多个列排序	GET_MULTI_COLUMN_SORT

（6）根据另一个字段的值对一个字段结合排序

如果某字段本身不返回所需的结果，则可以指定该字段使用当前字段进行结合排序。要

做到这一点，可以使用接口类 IF_SALV_WD_COLUMN_SERVICE_REF（实施类 CL_SALV_WD_COLUMN）的方法，见表 6-57。

表 6-57

功　　能	方　　法
指定其他字段的字段名	SET_SORT_FIELDNAME
获取其他字段的字段名	GET_SORT_FIELDNAME

6.4.2　过滤器

使用过滤器，可以限制 ALV 输出显示的数据记录。要做到这一点，可以在一个特定的字段中指定条件的数据记录，以满足要显示或过滤掉数据。

可以对过滤器对象进行以下设置：

- 创建、获取或删除字段中的过滤条件。
- 设定筛选条件。
- 忽略大小写。
- 设定一列禁止过滤。
- 使用另一个字段的值过滤。
- 在工具栏上的“过滤器”设置页面中隐藏或显示过滤标签页。

（1）创建、获取或删除字段中的过滤条件

不像排序条件，程序员可以为每个字段创建无限的过滤条件。

注：程序员只可以对没有禁止的列创建过滤条件。

过滤是 ALV 输出中一个字段的属性。创建、获取或删除字段的过滤条件，可以使用接口类 IF_SALV_WD_FILTER（实现类 CL_SALV_WD_FIELD）的方法，见表 6-58。

表 6-58

功　　能	方　　法
获取特定的过滤器条件	GET_FILTER_RULE
设定一个字段的所有过滤条件	GET_FILTER_RULES
创建一个过滤条件	CREATE_FILTER_RULE
删除一个特定的过滤器条件	DELETE_FILTER_RULE
删除所有字段中的过滤条件	DELETE_FILTER_RULES

一个对象的类CL_SALV_WD_FILTER_RULE代表一个字段的过滤条件。

注：如果用一个新的结构分配一个新的数据表到ALV输出，则所有字段的所有过滤器条件会被自动删除。

（2）设置筛选条件

过滤器的条件包括以下内容：

1）检查行的比较值。比较值可以是一个单独的值或范围。用户角度输入的是下限的值

（LOW_VALUE），作为一个范围，用户需输入一个上限的值（HIGH_VALUE）。

2）比较字段值和比较值大小操作。使用这个规范的，用户可以指定单元格的值和比较值之间的关系（如大于、小于或等于）。

3）包含或排除。使用这个规范的，可以指定是否显示不符合条件的行。

要更改设置的过滤条件，可以使用类 CL_SALV_WD_FILTER_RULE 方法，见表 6-59。

表 6-59

功　　能	方　　法
指定比较值（下限值）	SET_LOW_VALUE
取得比较值（下限值）	GET_LOW_VALUE
指定比较范围的上限值	SET_HIGH_VALUE
取得比较范围的上限值	GET_HIGH_VALUE
指定比较操作	SET_OPERATOR
获得比较操作	GET_OPERATOR
指定包含或排除	SET_INCLUDED
取得包含或排除	GET_INCLUDED

（3）忽略大小写

默认情况下，当过滤条件查找数据记录时，过滤器区分大小写。当用户输入比较值时必须完全正确地输入值，以获得所需的结果。

可以设置忽略大小写。要做到这一点，可以使用接口类 IF_SALV_WD_FIELD_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-60。

表 6-60

功　　能	方　　法
忽略大小写	SET_FILTER_IGNORING_CASE
检查是否忽略大小写	IS_FILTER_IGNORING_CASE

（4）设定一列禁止过滤

可以明确地禁止某个字段的过滤。这具有以下效果：

- 如果 ALV 输出被过滤，则过滤器行会自动显示，可以看到当前为每列设置的过滤。用户可以快速输入过滤条件。
- 如果不允许在一个字段中过滤，则过滤器中的行的相应单元格是空的并且不可输入。
- 在“设置”对话框中，“筛选器”选项卡页面中的相应列不再可用。
- 如果在应用程序中定义了这一字段的过滤条件，则过滤条件对 ALV 输出没有影响。

要禁止某个字段的筛选，可使用接口类 IF_SALV_WD_FILTER（实现类 CL_SALV_WD_FIELD）的方法，见表 6-61。

表 6-61

功　　能	方　　法
不允许字段过滤	SET_FILTER_ALLOWED
检查字段是否允许过滤	IS_FILTER_ALLOWED

（5）使用的另一个字段的值过滤

可以指定一个字段，根据该字段对当前字段进行过滤。如果该字段不能返回所需的结果，或者需要过于复杂的输入，则可以使用接口类 IF_SALV_W D_COLUMN_SERVICE_REF（实施类 CL_SALV_WD_COLUMN 的）的方法，见表 6-62。

表 6-62

功　　能	方　　法
指定字段名称的另一字段	SET_FILTER_FIELDNAME
获取另一字段的字段的名称	GET_FILTER_FIELDNAME

6.4.3 计算（合计）

可以对数值类型数据进行计算：生成聚合条件，计算的结果显示在一个单独的结果行中。

一般对一个字段的所有值使用聚集计算的过程中，也能够获得中间结果。要做到这一点，必须对 ALV 输出按一定条件排序，根据排序结果将行分组。

可以进行以下聚合设置：

- 创建、获取和删除聚合规则。
- 设置聚合。
- 生成中间结果。
- 隐藏或显示聚合计算或中间结果的接口元素。

可以为一个字段创建聚合规则中的最大值。聚合是 ALV 输出中一个字段的属性。创建、获取或删除字段的汇聚规则，使用接口类 IF_SALV_WD_AGGR（实施类 CL_SALV_WD_FIELD）的方法，见表 6-63。

表 6-63

功　　能	方　　法
取得聚合规则	GET_AGGR_RULE
创建汇总规则	CREATE_AGGR_RULE
删除汇总规则	DELETE_AGGR_RULE

一个字段的聚合规则表示为类CL_SALV_WD_AGGR_RULE的一个对象。

注：如果以一个新的结构分配ALV输出一个新的数据表，则各个字段的聚合规则被自动删除。

6.4.3.1 聚合设定

可以对字段值的计算进行以下设置：

- 设置聚合类型。
- 设置结果行的位置。
- 禁止某个字段的聚合。

（1）设置聚合类型

下面的计算（聚集）类型可供选择：

- 总计（Total）：所有的字段值加在一起。
- 最小值（Minimum）：确定字段中的最小值。
- 最大值（Maximum）：确定字段中的最大值。
- 平均值（Average）：确定所有字段值的几何平均值。

要定义计算类型，使用类的 CL_SALV_WD_AGGR_RULE 的方法，见表 6-64。

表 6-64

功　　能	方　　法
设置计算类型	SET_AGGREGATION_TYPE
取得计算类型	GET_AGGREGATION_TYPE

除了上面列出的计算类型，也可以决定数据记录的总数。测试结果将显示在结果中排在第一个可用的列中。由于该设置会影响整个 ALV 输出，因此可以使用接口类 IF_SALV_WD_FIELD_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-65。

表 6-65

功　　能	方　　法
显示数据记录的数目	SET_COUNT_RECORDS_ENABLED
检查是否显示数据记录的数量	IS_COUNT_RECORDS_ENABLED

（2）设置结果行的位置

在 ALV 输出中可以定义是否显示高于或低于该行中所包含的计算中的计算结果行。可以使用接口类 IF_SALV_WD_FIELD_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-66。

表 6-66

功　　能	方　　法
将结果行放在数据记录之前	SET_AGGR_BEFORE_ITEMS
检查结果行是否在数据记录之前	IS_AGGR_BEFORE_ITEMS

（3）禁止某个字段的聚合

默认情况下，各个数值类型字段的数据都可以聚合。如果需要，程序员可以禁止某个字段的聚集。这具有以下效果：

- “设置”对话框中没有对该列计算的标签页。
- 即使在应用程序中定义该字段的汇总规则，计算也不进行。

要禁止字段的聚集功能，使用接口类 IF_SALV_WD_AGGR（实施类 CL_SALV_WD_FIELD）的方法，见表 6-67。

表 6-67

功　　能	方　　法
禁止聚集	SET_AGGREGATION_ALLOWED
检查是否允许聚集	IS_AGGREGATION_ALLOWED

6.4.3.2　中间结果

默认情况下，字段中的所有值在聚集过程中都可以进行计算。同时，聚集还可以生成中

间结果。当生成中间结果时，将数据记录包含的值进行分组生成中间结果，并将中间结果显示在一个单独的结果行。

要生成中间结果，必须进行以下设置：

- 指定哪些字段值要计算中间结果，可以创建相关字段的聚合条件。总的结果显示在结果行。
- 指定哪些数据记录包含一个中间结果，将数据记录分组，在 ALV 输出对分组的字段排序，该字段包含在中间结果准则中。
- 要产生的中间结果，需将字段进行聚集，该字段包含在中间结果的规则中。
- 要显示中间结果，需在显示屏上激活中间结果显示。

注：产生中间结果的最常用的方法是使用小计。中间结果可用于所有聚合类型的中间结果。

（1）中间结果关联操作

- 生成中间结果。
- 显示中间结果。
- 指定级别下的中间结果。
- 设置结果行的位置。
- 禁用产生中间结果。

（2）先决条件

- 已经创建了至少一个字段可聚合的聚合条件。
- 该字段中间结果的准则不能被合并，包含一个字母数字类型数据的字段（用于分组）。

（3）生成中间结果

要在一个字段已经具有的聚合条件下产生中间结果，用另一列的字段创建一个排序条件。

在这种条件下，可以指定是否要生成中间结果。要做到这一点，可以使用类 CL_SALV_WD_SORT_RULE 的方法，见表 6-68。

表 6-68

功　　能	方　　法
生成中间结果	SET_GROUP_AGGREGATION
检查是否显示中间结果	GET_GROUP_AGGREGATION

（4）显示中间结果

已经完成所有设置后，要显示生成的中间结果，必须激活这些中间结果的显示。要做到这一点，可以使用接口类 IF_SALV_WD_FIELD_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-69。

表 6-69

功　　能	方　　法
显示中间结果	SET_GROUP_AGGR_DISPLAYED
检查是否显示中间结果	GET_GROUP_AGGR_DISPLAYED

（5）指定级别下的中间结果

如果已经定义了多个字母数字字段的中间结果，则该结果在多个级别小计：中间结果分层显示，根据级别并标有特定数目的点。用户可以使用这些点显示和隐藏中间结果的条目。在应用程序中，可以显示或隐藏一个或多个级别的小计条目。

也可以折叠中间结果：在这一过程中，可以隐藏所有数据记录和任何相应低级别的小计。只有结果行小计的最高级别结果和总计结果行保持可见。

对于这些功能，可以使用接口类 IF_SALV_WD_FIELD_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6−70。

表 6−70

功　能	方　法
指定显示最高级别小计	SET_EXPAND_LEVEL
获取当前显示小计级别	GET_EXPAND_LEVEL
折叠到中间结果	SET_GROUP_AGGR_COLLAPSED
检查数据是否被折叠成中间结果	GET_GROUP_AGGR_COLLAPSED

（6）禁用产生中间结果。

默认情况下，某个字段合计的中间结果被自动显示在 ALV 输出中。所有排序字母数字类型数据的字段被解释为可能的中间结果的计算准则，从而提供中间结果。

可以禁止特定字段中间结果的产生。如果应用程序在这一字段产生中间结果，则这些设置对 ALV 输出没有影响。

注：用户可以指定想要的列，以显示中间结果。如果是禁止生成中间结果的字段，则没有显示给用户的特定列。

要禁止字段产生中间结果，使用接口类 IF_SALV_WD_SORT（实现类 CL_SALV_WD_FIELD）的方法，见表 6−71。

表 6−71

功　能	方　法
禁止生成中间结果	SET_GROUP_AGGREGATION_ALLOWED
检查是否允许生成中间结果	IS_GROUP_AGGREGATION_ALLOWED

6.4.4　配置和个性化 ALV 输出

可以配置 Web Dynpro 应用，因此在设计时指定呈现给用户工作时的细节显示，而有些细节是不能在设计时设定的。此外，管理员和用户可以在运行时生成和保存自己的设置，调整应用程序的定制和个性化。

在 ALV 组件中也有各种选项修改了一组标准的属性，用户可以根据自己的业务需求为自己提供合适的变式。然而，与其他的 Web Dynpro 环境不同，用户不可以在设计时使用此配置。在设计时，有可能没有指定涉及配置的结构。出于这个原因，在一般情况下不能使用应用程

序配置工具来配置 ALV 输出。

（1）个性化配置和自定义 ALV 输出

1）个性化：ALV 输出的功能之一，提供给用户设定视图。在视图中用户可以保存列结构、排序条件、筛选条件和各种显示选项信息。这意味着用户可以使用其设置选项始终显示 ALV 输出。然而，这些配置视图只提供给创建它的用户，其他用户无法看到这些视图。

2）配置：应用程序开发人员使用 ALV 配置模型中的类及方法既不能创建也删除视图。但是，他们可以在特殊模式下在 ALV 组件中创建无限数量的视图，并交付这些视图。应用程序开发人员可以将视图保存在配置数据记录中，通过 ALV 编程将其传输和交付。

3）定制（自定义）：在客户的网站中，管理员可以改变他们所提供的视图，或插入自己的视图，然后将这些视图提供给所有用户客户端。

注：配置和定制，必须启动应用程序分别需要的模式，并在运行时进行必要的设置。

（2）视图和 ALV 配置模型的数据结构

所有设置，用户保存在 ALV 输出的视图中，该设置涉及一个特定数据结构的字段的 ALV 配置模型。例如，如果用户加载另一种结构时，它可能不再应用视图设置。

因此，必须确保系统能够唯一地分配，以便应用程序和数据结构同时加载。用一个唯一的密钥（KEY）标记数据结构。用户保存了所有视图，给出了这样的识别码。

程序员可以使用 ALV 组件接口控制器的方法 GET_CONFIG_DATA 得到一个视图所有的设置。

6.4.5 输出

用户可以在两个不同的文件格式下导出正在显示的 ALV 输出：

- 使用 Microsoft Excel
- 使用 Adobe Acrobat

此外，用户还可以把 ALV 输出输出到 BEx 分析仪。但是对结果不能做任何进一步的规范。

6.4.5.1 Microsoft Excel

从 ALV 输出生成 Excel 表，用户只需从工具栏选择相应的按钮即可。

在应用程序中，可以定义是否将 ALV 输出以外的信息复制到 Excel 文件中，如：

- 页眉和页脚区域的设计对象。
- 结果计算行。

注：目前，Excel格式支持版本MHTML和2000/97的MHTML。要设置想要的格式，执行报表SALV_BS_ADMIN_MAINTAIN。

（1）复制设计对象到 Excel 文件

如果已经定义了一个 ALV 输出的页眉或页脚的设计对象，则可以指定设计对象的内容是否复制到 Excel 文件中。把设计对象复制到 Excel 文件可以使用接口类 IF_SALV_WD_EXPORT_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-72。

表 6-72

功　　能	方　　法
复制设计对象为 Excel 页眉	SET_EXPORT_NO_TOL
检查是否复制设计对象到 Excel 页眉	GET_EXPORT_NO_TOL
复制设计对象为 Excel 页脚	SET_EXPORT_NO_EOL
检查是否复制设计对象到 Excel 页脚	GET_EXPORT_NO_EOL

（2）复制用于计算的结果行到 Excel 文件

可以定义合计结果行或中间小计结果行是否应该复制到 Microsoft Excel 中。可以使用接口类 IF_SALV_WD_EXPORT_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-73。

表 6-73

功　　能	方　　法
将结果行复制到 Excel 中	SET_EXPORT_NO_SUMS
检查结果行是否会被复制到 Excel 中	GET_EXPORT_NO_SUMS

6.4.5.2　Adobe Acrobat

用户可以打印当前显示的 ALV 输出。要做到这一点，用户可以选择工具栏上的按钮（Print Version）。ALV 根据 ALV 输出的数据生成一个标准的 PDF 文件，启动 Adobe Acrobat（系统已安装该软件），并显示生成的 PDF 文件。

用户可以使用“设置”对话框设置相关显示的 PDF 文件。在应用程序中也有相同的选项。可定义预设的 PDF 文档：

- 定义纸张格式和对齐。
- 定义可打印区域的大小。
- 缩放列和行。
- 直接将输出发送到打印机。
- 设置页眉和页脚。
- 复制设计对象到 PDF 文件。

也可以定义页眉或页脚的设计对象是否也应该显示在 PDF 文件中。

（1）定义纸张格式和对齐

可以指定 PDF 文档的文件格式。选择文件格式如下：

- DIN A4。
- Letter。

还可以定义是否要显示 PDF 文档为纵向或横向格式，可以使用接口类 IF_SALV_WD_PDF_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-74。

表 6-74

功　　能	方　　法
定义的纸张格式	SET_PAGE_SIZE
纸张格式	GET_PAGE_SIZE
设置纸张方向	SET_ORIENTATION
纸张方向	GET_ORIENTATION

（2）定义可打印区域的大小

可以定义 PDF 文档页面上的可打印区域的大小、指定页边距的宽度。也可以指定设置测量单位，可以使用接口类 IF_SALV_WD_PDF_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-75。

表 6-75

功　　能	方　　法
定义测量单位边距	SET_MARGINS_UNIT
取得测量单位边距	GET_MARGINS_UNIT
定义页脚、页眉、右边距和左边距的宽度	SET_MARGIN_BOTTOM SET_MARGIN_TOP SET_MARGIN_RIGHT SET_MARGIN_LEFT
取得页脚、页眉、右边距和左边距的宽度	GET_MARGIN_BOTTOM GET_MARGIN_TOP GET_MARGIN_RIGHT GET_MARGIN_LEFT

（3）缩放列和行

默认情况下，显示在 ALV 输出的 PDF 文件的列具有相同的宽度。如果在一个页面上没有足够的空间显示所有列，则该表被分割成两个或更多的页。每页同样容纳必要的行目。有不同的选项用于修改 PDF 文件的页面尺寸与列和行的尺寸：

- 减少列的宽度，使它们缩小到一个页面。
- 表的高度保持不变。
- 减少表中的宽度和高度，从而使整个表显示到一个页面上。
- 保持宽度和高度不变。
- 如果列分布在多个页面，则可以定义某些信息是否应该只出现在第一页或在随后的页面上重复。
- 如果列的宽度是不变的，则列标题显示在所有网页上。
- 如果列的宽度发生变化，则列标题在随后的页面上显示。
- 无论缩放：固定列重复的所有后续页。

缩放列和行，可以使用接口类 IF_SALV_WD_PDF_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-76。

表 6-76

功　　能	方　　法
定义 PDF 文档中的表缩放	SET_PAGE_LAYOUT
取得 PDF 文件中的表缩放	GET_PAGE_LAYOUT
如果页面宽度调整，则重复列标题	SET_REPEAT_HEADERS_FIT_H
如果页面宽度调整，则检查是否重复列标题	GET_REPEAT_HEADERS_FIT_H
相邻的页面，重复列标题	SET_REPEAT_HEADERS_WALLPAPER
相邻的页面，请检查是否重复列标题	GET_REPEAT_HEADERS_WALLPAPER
在每一页上重复固定的列	SET_REPEAT_KEY_COLUMNS
检查是否在每一页上重复固定的列	GET_REPEAT_KEY_COLUMNS

（4）直接将输出发送到打印机

可以选择要创建为 PostScript 文件，而不是一个 PDF 文件。PostScript 文件被直接发送到

选择的打印机。

注：如果没有在这里指定输出装置，则生成一个 PDF 文件，并显示在屏幕上。

设置即时打印，可以使用接口类 IF_SALV_WD_PDF_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-77。

表 6-77

功　　能	方　　法
设置 ALV 立即打印输出	SET_PRINT_IMMEDIATE
检查是否 ALV 立即打印输出	GET_PRINT_IMMEDIATE
设置打印机	SET_PRINTER
获取打印机	GET_PRINTER

（5）设置页眉和页脚

可以使用文本模块，为 PDF 文档创建页眉或页脚。页眉或页脚会出现在每一页中。这样做时，必须指定文本模块的位置（中间对齐、左对齐或右对齐）。

可以放置在所需位置的文本模块有：

- 无文字：当前位置没有被占用。
- 自定义文本：当前位置插入自定义文本。
- 当前日期：当前日期和时间
- 当前页。
- 页面 1 of ?

当前页数和总页数。

将文本模块放置于页眉和页脚中，可以使用接口类 IF_SALV_WD_PDF_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-78。

表 6-78

功　　能	方　　法
设置不同位置的自定义文本	SET_FOOTER_CENTER_FREETEXT SET_FOOTER_LEFT_FREETEXT SET_FOOTER_RIGHT_FREETEXT SET_HEADER_CENTER_FREETEXT SET_HEADER_LEFT_FREETEXT SET_HEADER_RIGHT_FREETEXT
取得不同位置的自定义文本	GET_FOOTER_CENTER_FREETEXT GET_FOOTER_LEFT_FREETEXT GET_FOOTER_RIGHT_FREETEXT GET_HEADER_CENTER_FREETEXT GET_HEADER_LEFT_FREETEXT GET_HEADER_RIGHT_FREETEXT
设置文本模块的不同位置	SET_FOOTER_CENTER SET_FOOTER_LEFT SET_FOOTER_RIGHT SET_HEADER_CENTER SET_HEADER_LEFT SET_HEADER_RIGHT
取得文本模块的不同位置	GET_FOOTER_CENTER GET_FOOTER_LEFT GET_FOOTER_RIGHT GET_HEADER_CENTER GET_HEADER_LEFT GET_HEADER_RIGHT

（6）复制设计对象到 PDF 文件

如果在 ALV 输出的页眉或页脚中已经定义了一个设计对象，则可以定义该设计对象的内容是否也应该被复制到 PDF 文件，可以使用接口类 IF_SALV_WD_PDF_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-79。

表 6-79

功　　能	方　　法
复制设计对象为 PDF 标题	SET_EXPORT_NO_TOL
检查是否复制设计对象为 PDF 标题	GET_EXPORT_NO_TOL
复制设计对象为 PDF 页脚	SET_EXPORT_NO_EOL
检查是否复制设计对象为 PDF 页脚	GET_EXPORT_NO_EOL

6.5　功能、交互和事件

ALV 输出的不只是给用户用来显示表结构数据。用户还可以用各种方式与 ALV 输出列表交互。可使用下列选项：

1）标准 ALV 功能。在工具栏“设置”对话框中，可以显示或隐藏各种 UI 元素，使用这些元素，用户可以访问 ALV 输出服务。

2）自定义，应用程序特定的功能。创建与该用户可以运行的功能设置或标准的 ALV 功能，从而有效地使用应用程序在工具栏上的 UI 元素。

3）互动式 UI 元素 ALV 输出表。在 ALV 输出表中，可以使用交互式 UI 元素（如列中的单元格编辑器和 ALV 输出的单元格）。有以两种类型的交互：

- 没有数据变化的交互。
- 有数据变化的交互。

UI 元素有用户操作的各种事件处理。

6.5.1　提供 ALV 功能

标准的 ALV 功能包括排序、筛选和计算。

除了标准的 ALV 功能外，可以使用用户自定义的、特定于应用程序的功能来显示 UI 元素。

可以进行以下功能设置：

- 提供标准的 ALV 功能。
- 创建、获取和删除功能。
- 指定 UI 元素。
- 在工具栏指定位置。
- 控制可见性和状态激活。
- 事件处理功能。

6.5.1.1　提供标准的 ALV 功能

可以选择不同的程序，为用户提供一个标准的 ALV 功能：

1）显示标准 ALV 功能：UI 元素的类型和使用的标签默认设置。根据标准 ALV 功能，UI 元素显示在“设置”对话框或在工具栏上。可以指定标准 ALV 功能是向用户提供相关的标准功能的 UI 元素。

2）提供工具栏的替代功能。如果想实现在工具栏上的 UI 元素与标准设置有所不同，则可添加不同类型的 UI 元素到工具栏并允许使用标准 ALV 功能，但隐藏相应的 UI 元素，然后生成一个用户定义的函数，并将方法分配到所需的标准 ALV 功能。

（1）显示标准 ALV 功能

如果使用 ALV 配置模型，则所有标准的 ALV 功能对象会自动创建。程序员只需要指定是否允许用户使用此相关功能即可。

注：默认情况下，所有的标准 ALV 功能都显示，但这里需要计算的标准功能是个例外（包括中间结果）。

可以在接口类 IF_SALV_WD_STD_FUNCTIONS（实现类 CL_SALV_WD_CONFIG_TABLE）找到所需的方法。所有这些方法都包含参数类型 ABAP_BOOL 值，见表 6-80。

表 6-80

功　能	UI 元素	注释/前提	方　法
		显示和隐藏列	
	列选择标签页		SET_COLUMN_SELECTION_ALLOWED
		Sorting	
	“排序”选项卡页面		SET_SORT_COMPLEX_ALLOWED
	通过单击列标题排序	列标题中的箭头图标	SET_SORT_HEADERCLICK_ALLOWED
		Filter	
	过滤器标签页		SET_FILTER_COMPLEX_ALLOWED
	工具栏上的过滤器按钮		SET_FILTER_FILTERLINE_ALLOWED
		Calculations	
	“计算”选项卡页面		SET_AGGREGATION_ ALLOWED
	所有元素的中间结果		SET_GROUP_AGGREGATION_ALLOWED
	计数表中条目		SET_COUNT_RECORDS_ALLOWED
		Display	
	显示标签页		SET_DISPLAY_SETTINGS_ALLOWED
	层次复选框和定义按钮		SET_HIERARCHY_ALLOWED
		Views	
	视图下拉列表框		SET_VIEW_LIST_ALLOWED
		Editable ALV	
	“插入行”按钮	如果 write-protection 属性设置无效只能显示	SET_EDIT_INSERT_ROW_ALLOWED
	“追加行”按钮		SET_EDIT_APPEND_ROW_ALLOWED
	“删除行”按钮		SET_EDIT_DELETE_ROW_ALLOWED
	“检查”按钮		SET_EDIT_CHECK_AVAILABLE
	“撤销”按钮		SET_EDIT_UNDO_ALLOWED

（续）

功　能	UI 元素	注释/前提	方　法
Export			
	导出到 Excel 中的下拉列表框		SET_EXPORT_ALLOWED
	导出到 BEx 分析仪的下拉列表框	如果配置相应的系统只能显示	SET_BEX_ANALYZER_ALLOWED
	打印版本按钮和标签页		SET_PDF_ALLOWED
Display As			
	显示方式为下拉列表框		SET_DISPLAY_AS_ALLOWED
	图形显示为下拉列表框中的条目	如果标准功能中的图形或 Excel 至少有一个被打开，则下拉列表框才是可见的	SET_GRAPHIC_ALLOWED
	Excel 中显示为下拉列表框中的条目		SET_EXCEL_INPLACE_ALLOWED
Send			
	发送按钮		SET_BI_BROADCASTING_ALLOWED

注：相关取得方法的名称以 IS_<method NAME>开始，而不是以 SET_<method NAME>开始。

（2）在工具栏提供替代功能

如果用户想在工具栏上使用标准设置以外的不同类型或标签 UI 元素，则允许使用标准 ALV 功能，但要隐藏相应的 UI 元素，然后生成一个用户定义的函数，并为其指定所需的标准 ALV 功能。

程序员可以用标准 ALV 功能替换如下 UI 元素：

- Send 下拉列表框与下拉列表框中的个别项目。
- Display As 下拉列表框与下拉列表框中的个别项目。
- Insert Row（插入行）、Append Row（追加行）和 Delete Row（删除行）按钮。
- Check（检查）按钮。
- Undo（撤销）按钮。
- Excel 和 Print Version（打印版本）按钮。
- Filter（过滤器）按钮、Define Filter（定义过滤）和 Delete Filter（删除过滤）功能。
- Settings（设置）按钮、Open Settings Dialog Box（打开设置对话框）和 Close Settings Dialog Box（关闭设置对话框）功能。
- Views（视图）下拉列表框。

要指定一个标准的 ALV 功能到一个自定义的函数，则可以使用类 CL_SALV_WD_FUNCTION 的方法，见表 6-81。

表 6-81

功　能	方　法
标准 ALV 功能分配到自定义的函数	SET_FUNCTION_STD
标准 ALV 功能，被分配到一个自定义的函数	GET_FUNCTION_STD

（3）系统设置导出到在 BEx 分析仪

如果用户想通过应用程序将 ALV 输出导出在 BEx 分析仪中，必须满足以下先决条件：

- 确保 BI 系统的安装。这意味着，BEx 分析仪也是可用的。
- 在 SAP NetWeaver 的定制中，单击“Application Server”→“Web Dynpro for ABAP”→“System-Wide Settings for the Web Dynpro ALV”命令，取消选择 Hide BEx Analyzer Export Function。

6.5.1.2　创建、获取和删除功能

如果使用 ALV 配置模型，则所有标准的 ALV 功能对象会自动创建。对象是类 CL_SALV_WD_FUNCTION_STD 的引用。

使用用户自定义的功能，在其功能函数中可以创建类 CL_SALV_WD_FUNCTION 的一个函数对象。根据需要，可以创建许多函数对象，并安排它们在工具栏上。

程序员可以进行以下设置：

- 创建一个用户定义功能函数的对象。
- 取得功能函数的对象。
- 设置和获取功能函数的信息。
- 删除一个用户定义功能的对象。

（1）创建一个用户自定义功能的对象

当创建一个用户定义函数时，程序员可以指定一个唯一的 ID（类型 STRING）。

要生成一个函数对象，使用接口类 IF_SALV_WD_FUNCTION_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-82。

表 6-82

功　能	方　法
创建用户定义的函数功能	CREATE_FUNCTION

（2）取得功能函数的对象

为了设置一个函数对象，必须首先调用一个适当的 GET 方法。程序员可以选择从工具栏上得到以下功能：

- 所有的标准 ALV 功能或特定的功能。
- 所有用户自定义的函数功能或特定的功能。
- 只有工具栏上面的标准的 ALV 功能。
- 所有工具栏上面的用户自定义的函数或特定的功能。

获取函数对象，可使用接口类 IF_SALV_WD_FUNCTION_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-83。

表 6-83

功　能	方　法
获取所有标准的 ALV 功能	GET_FUNCTIONS_STD
具体标准 ALV 功能	GET_FUNCTION_STD
获取所有用户定义的功能	GET_FUNCTIONS

（续）

功　　能	方　　法
获取特定的用户定义的功能	GET_FUNCTION
从左边的工具栏获取 ALV 所有标准功能	GET_FUNCTIONS_LEFT_STD
从左侧的工具栏获取所有用户定义的功能	GET_FUNCTIONS_LEFT
从左手侧的工具栏获取特定的用户定义的功能	GET_FUNCTION_LEFT
获取所有标准 ALV 的右侧工具栏的功能	GET_FUNCTIONS_RIGHT_STD
获取所有用户定义的函数右手边的工具栏	GET_FUNCTIONS_RIGHT
取得具体的右侧工具栏的用户定义的功能	GET_FUNCTION_RIGHT

（3）设置和获取功能函数的信息

可以检查工具栏的功能，以确定它是否是一个标准的 ALV 功能或用户定义的功能。

用户还可以指定以下属性：

1）ID。标准 ALV 功能的 ID 被自动创建。它包含了有关标准的 ALV 功能的信息。一个用户定义功能函数的 ID 是程序员在对象创建时指定的 ID。

2）Group。可以根据任何标准，使用组名划分功能。

对于设置和获取的信息，可以使用类 CL_SALV_WD_FUNCTION 或类 CL_SALV_WD_FUNCTION_STD 中的方法，见表 6-84。

表 6-84

功　　能	方　　法
功能型	GET_TYPE
功能 ID	GET_ID
分配功能的一组	SET_GROUP
功能被分配到组的名称	GET_GROUP

（4）删除一个用户定义功能的对象

程序员只能删除用户定义的功能函数的对象，不能删除标准 ALV 功能的对象。

要删除一个用户自定义的功能，需指定功能 ID。要做到这一点，可以使用接口类 IF_SALV_WD_FUNCTION_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-85。

表 6-85

功　　能	方　　法
删除用户定义的功能	DELETE_FUNCTION

6.5.1.3　预定义 Context 节点

一些自定义功能可以插入到工具栏中，当用户触发时会导致数据改变。因此，这些功能需要链接到应用程序 Context 节点。要做到这一点，必须定义一个外部环境映射到 ALV 接口控制器 Context 节点的 FUNCTION_ELEMENTS 上。为每个功能对应的 Context 节点生成一个合适的属性或子节点。

6.5.1.4　指定 UI 元素

可以为自定义功能使用下面 UI 元素：

- **Button**（CL_SALV_WD_FE_BUTTON）。
- **ButtonChoice**（CL_SALV_WD_FE_BUTTON_CHOICE）。
- **DropDownByIndex**（CL_SALV_WD_FE_DROPDOWN_BY_IDX）。
- **DropDownByKey**（CL_SALV_WD_FE_DROPDOWN_BY_KEY）。
- **InputField**（CL_SALV_WD_FE_INPUT_FIELD）。
- **LinkToAction**（CL_SALV_WD_FE_LINK_TO_ACTION）。
- **LinkToURL**（CL_SALV_WD_FE_LINK_TO_URL）。
- **ToggleButton**（CL_SALV_WD_FE_TOGGLE_BUTTON）。

对于个体 UI 元素之间的分隔线：

- **Separator**（CL_SALV_WD_FE_SEPARATOR）。

注：程序员可以在系统中的包 SALV_WD_CONFIG 中找到相关类。

指定 UI 元素的自定义功能的步骤如下：

- 生成一个功能对象。
- 生成工具栏上面列出的元素之一。
- 根据需要指定工具栏元素的属性。
- 分配工具栏元素到功能函数。

要指定一个合适的 UI 元素到一个函数对象，可以使用类 CL_SALV_WD_FUNCTION 中的方法，见表 6-86。

表 6-86

功　　能	方　　法
设置 UI 元素	SET_EDITOR
取得 UI 元素	GET_EDITOR

6.5.1.5　在工具栏中指定位置

可以以不同的方式指定功能的位置：

1）指定工具栏中 UI 元素对齐。可以指定功能元素在工具栏中的对齐方式是左对齐还是右对齐。

2）指定位置。可以指定在哪个位置显示功能。

要指定一个标准的 ALV 功能的位置，可以使用类 CL_SALV_WD_FUNCTION_STD 中的方法。

要指定一个自定义功能的位置，可以使用类 CL_SALV_WD_FUNCTION 中的方法，见表 6-87。

表 6-87

功　　能	方　　法
指定工具栏中的对齐方式	SET_ALIGNMENT
取得工具栏中的对齐方式	GET_ALIGNMENT
设置位置编号	SET_POSITION
取得位置编号	GET_POSITION

6.5.1.6 控制可见性和激活状态

可以指定是否以不同的方式使用自定义函数：

- 可以关闭一个单独的 UI 元素的功能。
- 可以隐藏一个单独的 UI 元素的功能。
- 可以关闭整个工具栏。
- 可以隐藏整个工具栏。

（1）激活和取消激活功能

如果用户不希望隐藏 UI 元素，要防止用户执行它的功能，则可以禁用 UI 元素。

要激活或禁用标准 ALV 功能的 UI 元素，需使用类 CL_SALV_WD_FUNCTION_STD 中的方法。

要激活或禁用的 UI 元素的自定义功能，可以使用 UI 元素类的方法，见表 6-88。

表 6-88

功　能	方　法
启用/禁用功能对象	SET_ENABLED
取得功能对象的状态	GET_ENABLED

（2）显示和隐藏功能

以下两个常量，可用于显示和隐藏功能。

1）VISIBILITY-VISIBLE：该系统显示的功能。

2）VISIBILITY-NONE：该系统隐藏的功能。

要显示或隐藏标准 ALV 功能的 UI 元素，可以使用类 CL_SALV_WD_FUNCTION_STD 中的方法。

要显示或隐藏 UI 元素的自定义功能，可以使用类 CL_SALV_WD_FUNCTION 中的方法，见表 6-89。

表 6-89

功　能	方　法
显示/隐藏功能对象	SET_VISIBILITY
获得显示状态	GET_VISIBILITY

（3）显示、隐藏和禁用的工具栏

为了控制整个工具栏的激活状态或可见性，使用的接口类 IF_SALV_WD_FUNCTION_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-90。

表 6-90

功　能	方　法
激活/关闭工具栏	SET_ENABLED
获得活动状态	GET_ENABLED
显示/隐藏工具栏	SET_VISIBILITY
获得显示状态	GET_VISIBILITY

6.5.1.7 事件处理函数

（1）标准 ALV 功能事件

使用标准 ALV 功能，只能在两个特定的时间点处理：当用户执行一个功能处理或当用户完成一个功能处理。用户选择没有收到任何有关的标准 ALV 功能。

可以使用下面的事件来处理标准的 ALV 功能：

- ON_STD_FUNCTION_BEFO。
- ON_STD_FUNCTION_AFTE。

（2）自定义的功能事件

使用自定义功能，可以决定哪些功能是用户选择的。

事件 ON_FUNCTION 是可用来处理自定义的功能。

6.5.2 没有数据变化的交互与处理

并非在所有 ALV 输出中用户的操作会引起数据的变化。例如下面的操作，不会导致数据的变化：

- 改变行的选择（选择行）。
- 选择超链接、按钮及开关按钮。

程序员可以进行各种设置，以控制或处理这种交互。

6.5.2.1 选择行和列

可以决定用户是否可以一次性选择多个行或列。

- 能够选择列。使用选择的状态，程序员可以指定用户在同一时间是否可以选择多列。
- 能够选择行。使用选择类型，程序员可以指定用户在同一时间是否可以选择多行以及是否存在头选择（Lead Selection）。

注：用户还可以在写保护的 ALV 输出中选择行。如果关闭 ALV 输出的写保护，则无论选择类型用户都不可以选择任何行。

也可以屏蔽用户选择特定的单元格或所有单元格。

（1）设置选择状态

程序员可以分别为每列指定选择状态。可以使用下面的选择状态：

- 未选择：任何现有的选择将被删除。
- 选择：该列被选中。
- 不可选：用户不可以选择该列。

要解决列选择，可以使用类 CL_SALV_WD_COLUMN 的方法，见表 6-91。

表 6-91

功　　能	方　　法
指定选择状态	SET_SELECTION_STATE
取得选择状态	GET_SELECTION_STATE

（2）设置选择类型

可以使用下面的选择类型：

- 自动：从 Context 节点设置。
- 无选择（NONE）：在行的最右端没有任何用于选择行的按钮显示。
- 单选择（SINGLE）：在行的最右端显示用于选择行的按钮。用户只能选择单列。该行被显示为头选择。如果用户选择了另一行，则头选择也被改变。
- 无头选择的单选择（SINGLE_NO_LEAD）：在行的最右端显示用于选择行的按钮。用户只能选择一个单列。如果用户选择了另一行，则选择将被改变。ALV 输出不含头选择。
- 多行（MULTI）：在行的最右端显示用于选择行的按钮。用户可以通过按住〈Ctrl〉键的同时选择多行。选择的第一行为头选择。
- 不含头选择的多行（MULTI_NO_LEAD）：在行的最右端显示用于选择行的按钮。用户可以通过按住〈Ctrl〉键的同时选择多行。ALV 输出不含头选择。

注：程序的 Context 节点来指定有多少个数据记录，可以选择使用 Selection 属性。

如果 Selection 属性值 1：1，则必须选择一个条目。如果该值是 0：n，它是可以不选择任何条目，或根据需要选择多条。

使用选择类型时，用户不能选择 Context 节点中定义的数字。如果尝试做到这一点，运行时发生错误，则应用程序终止。

例如：如果指定了 1：1，就不能使用选择类型 MULTI 和 MULTI_NO_LEAD 的。

要定义选择类型，使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-92。

表 6-92

功　能	方　法
指定选择类型	SET_SELECTION_MODE
取得选择类型	GET_SELECTION_MODE

（3）事件处理

默认情况下，在列表区域中有两种不同的动作响应用户选择的事件：

- ON_LEAD_SELECTION：该动作改变用户的头选择。如果用户改变选择，则没有事件被触发。
- ON_SELECT：该动作用于改变选择或头选择。

使用方法 IF_SALV_WD_TABLE_SETTINGS~SET_ON_SELECT_ENABLED，系统切换到事件 ON_SELECT。

注：如果将系统切换到 ON_SELECT，即停用 ON_LEAD_SELECTION 事件的事件处理程序。

（4）屏蔽选择

可以屏蔽用户选择 ALV 输出行。

注：与 NONE 选择类型相反，按钮不隐藏，它们只停用。这能够阻止用户选择行。

程序员可以决定是否要屏蔽所有 ALV 输出行，或者只是屏蔽特定行的选择。

要做到这一点，使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-93。

表 6-93

功　　能	方　　法
禁止选择所有 ALV 输出行	SET_ROW_SELECTABLE
检查是否不允许选择所有行	GET_ROW_SELECTABLE
禁止选择 ALV 输出中的特定行	SET_ROW_SELECTABLE_FIELDNAME
获取字段名，可以控制是否可以选择单独的行	GET_ROW_SELECTABLE_FIELDNAME

6.5.2.2　使用超链接、按钮和切换按钮

在 ALV 输出的单元格中可以使用 UI 元素，允许用户来触发某些动作。下面的 UI 元素并不会自动导致数据的变化：

- 超链接 LINK_TO_ACTION（UI 元素 LinkToAction）。
- 超链接 LINK_TO_URL（UI 元素 LinkToURL）。
- 按钮（UI 元素 Button）。
- 切换按钮（UI 元素 Toggle Button）。

注：以上所列的交叉引用，引用一般可用的 Web Dynpro 对象。指定的类名，是使用表 6-94 中列出的 ALV 配置模型的类名。

表 6-94

UI 元素	类
超链接 LINK_TO_ACTION	CL_SALV_WD_UIE_LINK_TO_ACTION
超链接 LINK_TO_URL	CL_SALV_WD_UIE_LINK_TO_URL
按钮	CL_SALV_WD_UIE_BUTTON
开关按钮	CL_SALV_WD_UIE_TOGGLE_BUTTON

可以使用事件 ON_CLICK 来处理以下 UI 元素的用户动作：

- 超链接 LINK_TO_ACTION。
- 按钮。
- 切换按钮。

6.5.3　可编辑 ALV 的互动处理

可以使用 ALV 输出的列显示 UI 元素，用户可以用它来改变显示的数据。

用户还可以在特定的位置插入行，把它们添加到 ALV 输出的底部，并将其删除。

这些操作之前，ALV 输出必须停用写保护。

还必须定义时间，在该时间系统会检查更改过的数据是否是正确的。

6.5.3.1　写保护及激活

可以在 ALV 输出中显示 UI 元素，可以改变 ALV 输出中的数据。不过，也有不同的方法，

可以屏蔽用户在这些 UI 元素上改变数据。可以进行以下设置：

- 控制 ALV 输出的写保护。
- 激活和取消激活 ALV 输出。
- 禁用单独的 UI 元素。

（1）控制 ALV 输出的写保护

默认情况下，ALV 输出是写保护的，要允许用户更改数据，则必须先删除该写保护。写保护有以下影响：

- UI 元素（在单元格中，而不是在工具栏中），允许要改变的数据不能被使用。
- 如果 ALV 输出被写保护，在工具栏上的标准 ALV 功能（如插入、添加、删除行和检查数据）被隐藏。

注：如果用户删除 ALV 输出的写保护，则系统为 ALV 输出使用指定的颜色。用户不能再影响用于 ALV 输出的颜色。

要打开或关闭写保护，可以使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-95。

表 6-95

功　　能	方　　法
打开/关闭写保护	SET_READ_ONLY
检查是否写保护开关 ON / OFF	GET_READ_ONLY

（2）激活和取消激活 ALV 输出

默认情况下，ALV 输出被激活。在 ALV 输出的所有交互的 UI 元素，在工具栏和表中的部分均可操作。

要启动或关闭 ALV 输出，使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-96。

表 6-96

功　　能	方　　法
启动或停用 ALV 输出	SET_ENABLED
检查 ALV 输出是否被激活	GET_ENABLED

6.5.3.2　添加和删除行

可以在工具栏上为用户提供按钮，在 ALV 输出不同的地方添加或删除数据记录。提供以下功能：

- 插入行。如果用户没有选择行，则创建新行在第一行。否则，它是建立在选定的一行或多行位置。
- 追加行。一个新行追加到 ALV 输出底部。
- 删除行。选定的行会被删除。

用户可以插入或删除的行的数目取决于它已经选择了多少行。可以使用选择类型来定义

用户可以一次性选择多少行。

注：如果已删除写保护的 ALV 输出，则这些按钮是唯一可见的。

如果没有选择 No Selection Possible，则用户不可以选择任何行。因此，用于插入和删除行的按钮被隐藏。

（1）事件处理

事件 ON_DATA_CHECK 包含插入或删除的行索引的参数。

（2）整页追加输入就绪行

用户可以切换 SAP 列表查看器（ALV）到海量数据的模式。从而自动创建一个空的页，输入就绪行在 ALV 输出结束的地方。用户不必单独插入每一个新的行，但可以在 ALV 输出结束的地方填充任意数目的行。

ALV 输出切换海量数据的模式，使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-97。

表 6-97

功　　能	方　　法
对大容量数据的开/关切换模式	SET_EDIT_MODE
检查是否对大容量数据的模式切换	GET_EDIT_MODE

6.5.3.3　更改单元格的内容

用户可以使用下面的 UI 元素，直接改变在 ALV 输出数据：

- 输入框（InputField）。
- 复选框（CheckBox）。
- 切换按钮（Toggle button）。
- 三态复选框（TriStateCheckBox）。
- 下拉列表框（DropDownByKey）。

注：上面列出的交叉引用是指常见的 Web Dynpro 对象，不是指定的类名，而是 ALV 配置模型。

可以使用下面的类名，见表 6-98。

表 6-98

UI 元素	类
输入框	CL_SALV_WD_UIE_INPUT_FIELD
复选框	CL_SALV_WD_UIE_CHECKBOX
切换按钮	CL_SALV_WD_UIE_TOGGLE_BUTTON
三态复选框	CL_SALV_WD_UIE_CHECKBOX_TRI
下拉列表框	CL_SALV_WD_UIE_DROPDOWN_BY_KEY

注：用户使用 UI 元素之前，必须关闭 ALV 输出的写保护，还必须确保 ALV 输出被激活。

6.5.3.4 指定检查时间

在可编辑的 ALV 中，用户可以更改数据，添加或删除行。这些变化最初只存储在 ALV 组件的 Context 中。在预定义的时间内，系统的数据从 ALV 组件 Context 传输到应用程序的 Context 中，并检查为正确的数据类型的数据。这确保了后续行动被应用到最新的数据。

例如，当用户单击工具栏中的“设置”按钮使用 ALV 服务时，系统会自动检查数据。

可以指定其他时间用于 ALV 输出中的当前数据与应用程序同步：

- 当用户按下〈Enter〉键或触发系统的动作（DATA_CHECK_ON_CELL_EVENT）。

使用此设置时，应确保在最小的数据更改的状况下，避免由于频繁的检查对系统性能有负面影响。

- 用户选择检查（DATA_CHECK_ON_CHECK_EVENT）。默认情况下，在关闭写保护的 ALV 输出中显示这个按钮。该设置是有用的，如果用户一个一个处理多个数据记录时，不会由于频繁地检查对系统性能有负面的影响。
- 当触发系统中的检查。

（1）由用户指定检查时间

要指定用户动作触发数据检查的时间，使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-99。

表 6-99

功　能	方　法
设定时间	SET_DATA_CHECK
取得时间	GET_DATA_CHECK

（2）事件处理

当数据被检查时会触发事件 ON_DATA_CHECK。

6.5.3.5 刷新显示

默认情况下，对变化的数据检查后，ALV 输出会进行自动刷新。在列表中的数据被更新，并且对话框的设置被应用到数据中。

（1）自动刷新

系统自动更新显示，在下列情况下对话框中被应用的设置，用户不能改变这些机制：

- 用户插入行，把它们添加到列表末尾或删除它们。
- 应用程序应用于对数据的更改。
- 用户通过单击“应用”或“确定”按钮将对话框中的设置应用到的数据中。
- 用户排序或者过滤 ALV 输出。

（2）手动刷新

在用户更改现有的数据记录的情况下，可以影响 ALV 输出刷新是要么只影响到的数据，要么当前设置的服务施加到更改后的数据。这有两种方法中的每一个都有两个可能的设置：

- SET_REFRESH_ON_DATA_CHANGE：控制刷新，当用户更改的数据证实了这一点，按〈Enter〉键。
- SET_REFRESH_ON_DATA_CHECK：控制刷新，当用户更改的数据，并选择检查按钮。

下面的值是可能的：

- REFRESH_AND_APPLY_SERVICES：在列表中的数据被更新的设置对话框中的设置被应用到数据中。
- REFRESH_DATA_ONLY：在列表中的数据被更新。

6.5.4 为用户提供帮助

可以选择不同的方式为应用程序的用户提供帮助。可以在 ALV 输出中使用以下类型的帮助：

- 辅助功能说明（accessibility Description）。如果用户已激活辅助功能，则指定的文本会添加到快速信息（Quick Info）。这个描述是提供更多的信息到 ALV 输出或列。当用户将焦点移动到 ALV 输出或列时该描述被浏览器读取。
- 工具提示。

当用户将鼠标指针停留在工具栏上的 UI 元素或单元格上时，快速信息文本被显示。如果浏览器支持，则辅助功能说明会作为额外的文本自动添加到快速信息。

- 说明（Explanation）。在工具栏或列表中的各种 UI 元素的说明属性。如果用户切换到帮助模式下，则当用户将光标移到 UI 元素时，该属性的文本显示在一个盒子里。文本说明属性的显示独立于工具提示。
- Web Dynpro 输入帮助。

可以在 ALV 输出中使用下面的输入帮助：ABAP 字典搜索帮助、OVS 输入帮助、自定义输入帮助。

（1）辅助功能说明

辅助功能说明属性用于以下几个方面：

1）整个 ALV 输出。

辅助功能说明显示 ALV 输出的标题的工具提示。要显示它，可以使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-100。

表 6-100

功　　能	方　　法
指定辅助功能说明	SET_ACC_DESCRIPTION
取得辅助功能说明	GET_ACC_DESCRIPTION

2）列。

辅助功能说明显示列标题的工具提示。要显示它，可以使用类 CL_SALV_WD_COLUMN 的方法，见表 6-101。

表 6-101

功　　能	方　　法
指定辅助功能说明	SET_ACCESSIBILITY_DESCR
取得辅助功能说明	GET_ACCESSIBILITY_DESCR

（2）工具提示

可以为 ALV 输出以下元素指定工具提示，见表 6-102。

表 6-102

工 具 提 示	类	方 法
对于 ALV 输出的标题	IF_SALV_WD_TABLE_SETTINGS	SET_TOOLTIP 和 GET_TOOLTIP
对于用户定义的函数	CL_SALV_WD_FE_<UI element>	
对于单元格编辑器	CL_SALV_WD_UIE_<UI element>	

- ALV 输出标题。
- 列标题。
- 用户自定义功能（通过使用的 UI 元素）。
- 单元格编辑器（通过使用的 UI 元素）。

（3）说明

可以指定 ALV 输出中下列内容的说明：

1）列。

2）用户自定义功能（通过使用的 UI 元素）。

- 按钮。
- 下拉列表框（索引和键）。
- 输入字段。

3）单元格编辑器（通过使用的 UI 元素）。

- 按钮。
- 复选框和三态复选框。
- 下拉列表框（索引和键值）。
- 输入字段。
- 文件上传。

说明的类和方法见表 6-103。

表 6-103

工 具 提 示	类	方 法
对于 ALV 输出的标题	CL_SALV_WD_COLUMN	SET_EXPLANATION GET_EXPLANATION
对于用户定义的函数	CL_SALV_WD_FE_BUTTON	
	CL_SALV_WD_FE_DROPDOWN_BY_IDX CL_SALV_WD_FE_DROPDOWN_BY_KEY	
	CL_SALV_WD_FE_INPUT_FIELD	
对于单元格编辑器	CL_SALV_WD_UIE_BUTTON	
	CL_SALV_WD_UIE_CHECKBOX	
	CL_SALV_WD_UIE_CHECKBOX_TRI	
	CL_SALV_WD_UIE_DROPDOWN_BY_IDX CL_SALV_WD_UIE_DROPDOWN_BY_KEY CL_SALV_WD_UIE_FILE_UPLOAD CL_SALV_WD_UIE_INPUT_FIELD	

6.6 接口控制器的方法和事件

接口控制器的 ALV 组件中可以使用下面的方法：

- DATA_CHECK。
- GET_CONFIG_DATA。
- GET_MODEL。
- GET_MODEL_EXTENDED。
- SET_DATA。

还可以使用 ALV 组件的接口控制器的下列事件：

- ON_AFTER_CONFIG。
- ON_CELL_ACTION。
- ON_CLICK。
- ON_DATA_CHECK。
- ON_FUNCTION。
- ON_LEAD_SELECT。
- ON_SELECT。
- ON_STD_FUNCTION_AFTE。
- ON_STD_FUNCTION_BEFO。

6.6.1 DATA_CHECK

如果用户在 ALV 输出中从 ALV 组件 Context 节点上已更改数据，则将数据传输到应用程序的 Context 节点上时，程序员可以使用可编辑 ALV 的 DATA_CHECK 方法。

SAP 强烈建议，在应用程序 Context 节点绑定新的或已更改的数据前（如使用 BIND_TABLE）使用这种方法。

往往条目要经历几个检查以确保正确性，如数据类型和格式是否正确。

注：程序员可以决定系统何时会自动调用该方法。

- 当用户单击“检查”按钮。
- 当用户选择返回或单击另一个单元格。

要做到这一点，使用方法 IF_SALV_WD_TABLE_SETTINGS~SET_DATA_CHECK。

6.6.2 GET_CONFIG_DATA

用户可以保存设置视图中的排序、筛选、执行计算等。方法 GET_CONFIG_DATA 返回该用户已经保存当前应用程序的视图的信息：

- 哪些视图已保存到应用程序了？这些视图名称是什么？
- 视图现在是否激活（已加载）？这一视图名称是什么？哪些设置已被保存到这个视图了？
- 作为初始视图，用户定义了特定视图没有？这一视图名称是什么？哪些设置已被保存到这个视图？

- 哪些视图属于哪个结构？（只适用于当几个结构使用 SET_DATA 方法装入一个 ALV 组件时）产生独特的配置项，并将它们分配到独立结构。

在 configuration key 的例子中，可以看到如何生成和分配配置密钥。

分配结构的密钥，必须符合以下条件：

- 识别应用程序中 ALV 输出的显示。可以使用 S_PARAM_OUT 中的字段 USAGE_PATH 用于此目的。
- 它确定了在应用程序中的 ALV 输出。如果有多个 ALV 输出显示在应用程序中，则它们必须能够区分彼此。要做到这一点，可以使用字段 ALV_COMPONENT_USAGE。
- 使用方法 SET_DATA。它标识在 ALV 输出中显示的结构。

要做到这一点，可以使用任何唯一的字符串。

以下示例方法 SET_ALV_CONFIG_ID 中显示了所需要的配置密钥的生成和分配的步骤。

```
METHOD SET_ALV_CONFIG_ID.
[ ... ]

  DATA:
     L_REF_IF_CONTROLLERTYPE REF TO IWCI_SALV_WD_TABLE,
     LR_NODE                 TYPE REF TO IF_WD_CONTEXT_NODE,
     OWN_KEY                 TYPE STRING.
*... GET RELEVANT NODE OF THE APPLICATION CONTEXT
  LR_NODE = WD_THIS->APPL_GET_DATA_NODE( ).

*... DETERMINE NAME FOR KEY DEPENDING ON THE APPLICATION CONTEXT
  OWN_KEY = ‘ALV1’.

*... SET NODE OF THE APPLICATION CONTEXT
  L_REF_IF_CONTROLLER->SET_DATA( R_NODE_DATA = LR_NODE ).

[ ... ]

*... CREATE CONFIGURATION KEY CONSISTING OF
*   USAGE PATH, COMPONENT USAGE, OWN KEY
DATA:
     LS_PARAM_OUT TYPE IF_SALV_WD_TABLE=>S_TYPE_PARAM_CONFIG_OUT,
     LS_PARAM_IN  TYPE IF_SALV_WD_TABLE=>S_TYPE_PARAM_CONFIG_IN,
     L_KEY       TYPE STRING,
     L_KEY_32     TYPE STRING.

  LS_PARAM_IN-ACTION = IF_SALV_WD_TABLE=>KEY.
  LS_PARAM_OUT = L_REF_IF_CONTROLLER->GET_CONFIG_DATA( LS_PARAM_IN ).
  CONCATENATE
     LS_PARAM_OUT-USAGE_PATH
     LS_PARAM_OUT-ALV_COMPONENT_USAGE
     OWN_KEY
```

```
INTO
  L_KEY
SEPARATED BY '&'.

*... HASH CONFIGURATION KEY TO UNIQUE KEY OF 32 CHARS LENGTH
  TRY.
      L_KEY_32 = CL_RSMDS_HASH_UTILITIES=>TO_HASH_C32( L_KEY ).
      CATCH CX_RSMDS_INPUT_INVALID CX_RSMDS_INPUT_INVALID_TYPE.
  ENDTRY.

[ ... ]

*... SET NEW CONFIGURATION KEY
  LS_PARAM_IN-ACTION                          = IF_SALV_WD_TABLE=>SET.
  LS_PARAM_IN-CONFIG_KEY-CONFIG_TYPE          = '08'.
  LS_PARAM_IN-CONFIG_KEY-CONFIG_ID            = L_KEY_32.

  L_REF_IF_CONTROLLER ->GET_CONFIG_DATA( LS_PARAM_IN ).
```

6.6.3 GET_MODEL

方法 GET_MODEL 返回一个标准执行的 ALV 配置模型。因此，可以用于配置 ALV 输出，以满足用户的特定要求。

标准执行的ALV配置模型包含了对应于应用程序Context节点属性的所有字段和列对象。

注：可以使用 GET_MODEL_EXTENDED 方法，以确定是否需要字段对象或列对象。

如果用户使用应用程序配置 ALV 输出，或者用户改变了 ALV 输出，则对 ALV 配置模型有以下影响：

- 改变相应字段的对象的配置过滤器、排序、聚集条件。
- 相应的列的对象被改变，如隐藏和显示列。

可以使用方法GET_MODEL调用当前状态的ALV输出与所有的服务、功能等的设置。通过这种方式，可以传输到另一个组件的一个特定的 ALV 输出的所有属性，并可以继续处理。

已经生成了一个 Context 节点的所有属性，将它与 ALV 组件外部环境进行映射或者使用方法 SET_DATA 完成映射。

6.6.4 GET_MODEL_EXTENDED

方法 GET_MODEL_EXTENDED 返回 ALV 配置模型。因此，可以用其配置 ALV 输出，以满足用户的特定要求。在标准执行的 ALV 配置模型中，检索使用 GET_MODEL 方法，决定是否需要另外字段对象的列，使用 GET_MODEL_EXTENDED 方法。

已经生成了一个 Context 节点的所有属性，将它与 ALV 组件外部环境进行映射或者使用方法 SET_DATA 完成映射。

6.6.5 SET_FOCUS

为了能够利用键盘使用 SAP 列表查看器（ALV），让用户在 ALV 输出的一行或一单元格上设置焦点。可以通过各种动作从 ALV 输出中删除焦点，然后使用方法 SET_FOCUS，将焦点设置在一个特定的行或单元格。

- 如果想将焦点设置到一行，则只使用索引参数。
- 如果想将焦点设置到一个单元格，则使用参数索引和列。

6.6.6 SET_DATA

使用方法 SET_DATA 有以下两个原因：

- 要传输应用程序 Context 数据到 ALV 组件，而不是使用外部环境映射。
- 在 ALV 输出中，要显示的数据来源于动态生成的 Context 节点。

注：如果随后要指定另一种结构到 ALV 输出中，则可以使用方法 SET_DATA。在这种情况下，不能在 WDDOINIT 方法或另一种标准方法中使用该方法。如果 ALV 输出已经显示一次，则可以只定义 ALV 组件的事件处理程序中的数据。

作为参数传输到方法 SET_DATA 的 Context 节点必须是已经存在的。这与该 Context 节点是否在 Web Dynpro 资源管理器中设计时生成的，还是程序在运行过程中自动生成无关。

6.6.7 ON_AFTER_CONFIG

目前不支持此事件。

6.6.8 ON_CELL_ACTION

默认情况下，可以使用两个不同的事件做出反应，用于在列表区域中与用户交互：

- ON_CLICK。如单击按钮或链接响应交互，不会导致数据的变化。
- ON_DATA_CHECK。在变化的数据时响应交互，如选中复选框或单击切换按钮，输入 InputFields 中的条目，或从下拉式列表框中选择结果。

用户可以设置 SAP 列表查看器（ALV），以便这两种情况下只有一个事件被触发：ON_CELL_ACTION 事件。要做到这一点，必须相应地设定系统。

如果转换系统到 ON_CELL_ACTION，则禁用 ON_CLICK 事件的事件处理程序。事件处理程序仍然对 ON_DATA_CHECK 事件是可用的。

将系统切换到 ON_CELL_ACTION，需使用接口类 IF_SALV_WD_TABLE_SETTINGS（实现类 CL_SALV_WD_CONFIG_TABLE）的方法，见表 6-104。

表 6-104

功　　能	方　　法
将系统切换到 ON_CELL_ACTION	SET_CELL_ACTION_EVENT_ENABLED
检查系统是否已经被切换至 ON_CELL_ACTION	IS_CELL_ACTION_EVENT_ENABLED

6.6.9　ON_CLICK

当用户单击 ALV 输出中单元格中的以下 UI 元素之一时，事件 ON_CLICK 被触发：

- 按钮。
- LinkToAction。
- ToggleButton。

注：如果方法 IF_SALV_WD_TABLE_SETTINGS~SET_CELL_ACTION_EVENT_ENABLED 设置为 false，则只触发 ON_CLICK，否则触发事件 ON_CELL_ACTION。

前提：已注册此事件。

6.6.10　ON_DATA_CHECK

ON_DATA_CHECK 事件被触发时，在一个可编辑 ALV 输出中检查已更改的数据：

- 该用户已插入或附加一个或多个行。
- 用户已经删除了一个或多个行。
- 用户已更改单元格中的数据。

可以进行数据校验时定义：

- 当用户从工具栏上单击检查按钮。
- 当用户选择回车或触发系统操作。
- 当执行方法 DATA_CHECK。

前提：已注册此事件。

6.6.11　ON_FUNCTION

当用户在应用程序工具栏上选择一个已经自定义创建的 UI 元素，特定功能被触发时的事件 ON_FUNCTION。

6.6.12　ON_LEAD_SELECT

当用户在 ALV 输出中更改头选择时，ON_LEAD_SELECT 事件被触发。

已经为 ALV 输出定义了一个选择类型，允许行选择和修改头选择。

6.6.13　ON_SELECT

（1）使用

当用户在 ALV 输出中更改头选择或进行任何选择时，触发事件 ON_SELECT。

（2）先决条件

- 已经注册此事件。
- 在 ALV 配置模型中激活事件 ON_SELECT。

可以使用方法 IF_SALV_WD_TABLE_SETTINGS~SET_ON_SELECT_ENABLED 激活该事件。

6.6.14 ON_STD_FUNCTION_AFTE

（1）使用

当用户执行下列操作之一时，触发事件 ON_STD_FUNCTION_AFTE：

- 用户单击设置对话框中的“复制”或“确定”按钮。
- 用户在工具栏上单击删除过滤器按钮。

注：可以使用该事件来确定用户执行的标准 ALV 功能。

（2）前提

已注册此事件。

6.6.15 ON_STD_FUNCTION_BEFO

（1）使用

当用户执行下列操作之一时触发事件 ON_STD_FUNCTION_BEFO：

- 用户单击工具栏中的“设置”按钮。
- 用户在工具栏上单击“过滤器”按钮。

注：可以使用该事件来确定用户执行的标准 ALV 功能。

（2）前提

已注册此事件。

6.7 在 Web Dynpro 应用中使用 ALV 的简单实例

1）在 ABAP 工作台中的编辑界面创建 Web Dynpro 组件。选择“Web Dynpro Comp./Intf”，在相应的文本框中输入程序的名称，如图 6-6 所示。

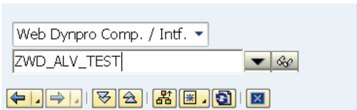


图 6-6

创建 Web Dynpro 组件如图 6-7 所示。

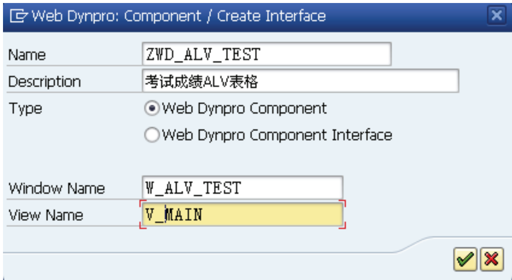


图 6-7

编辑 Web Dynpro 组件如图 6-8 所示。

Object Name	Description
ZWD_ALV_TEST	考试成绩ALV表格
COMPONENTCONTROLLER	考试成绩ALV表格----組件控制器
Component Interface	
INTERFACECONTROLLER	考试成绩ALV表格----組件控制器
Interface Views	
W_ALV_TEST	考试成绩ALV表格----窗体
Views	
V_MAIN	考试成绩ALV表格----视图
Windows	
W_ALV_TEST	考试成绩ALV表格----窗体

图 6-8

2）选择组件 ZWD_ALV_TEST，选择组件使用选项卡，添加组件使用如图 6-9 所示。

Web Dynpro Component ZWD_ALV_TEST Inactive/revised
Description 考试成绩ALV表格
Assistance Class
Created By BCUSER Created On 2013/02/26
Last Changed By BCUSER Changed On 2013/02/26
Original Lang. ZH Package ZWD_TRAINING
Accessibility Checks Active
Used Components | Implemented interfaces

Used Web Dynpro Components

Component Use	Component	Description of Component
SELECT_OPTIONS	WDR_SELECT_OPTIONS	
ALV_TABLE	SALV_WD_TABLE	

图 6-9

3）选择组件控制器，选择“Context”选项卡，创建 Context 节点如图 6-10 所示。

Component Controller COMPONENTCONTROLLER Active(revised)
Properties | Context | Attributes | Events | Methods
Controller Usage
Context COMPONENTCONTROLLER
CONTEXT
NODE_TEST
ZZCLASS
ZZYEAR
ZZSTDNO
ZZNAME
ZZTENKOK
ZZTENSAN
ZZTENRIK
ZZTENSHA

图 6-10

Context 节点属性设置如图 6-11 所示。

Change Node

Node Name	NODE_TEST
Interface Node	No
Input Element (Ext.)	No
Dictionary structure	ZZTEST_S
Cardinality	0..n
Selection	0..1
Init. Lead Selection	Yes
Singleton	No
Supply Function	
Mapping Path	

Add Attribute from Structure

图 6-11

4）在左侧的树形结构菜单上，按照以下路径双击组件控制器使用 INTERFACECONTROLLER_USAGE，如图 6–12 所示。

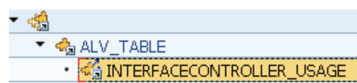


图 6–12

选择“Context”选项卡，将其下的 Context 节点 DATA 映射到组件控制器中的 Context 节点，如图 6–13 所示。

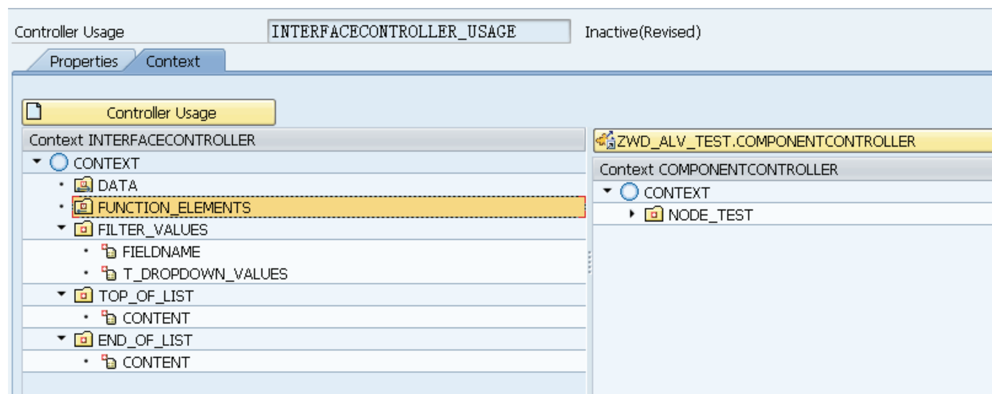


图 6–13

映射结果如图 6–14 所示。

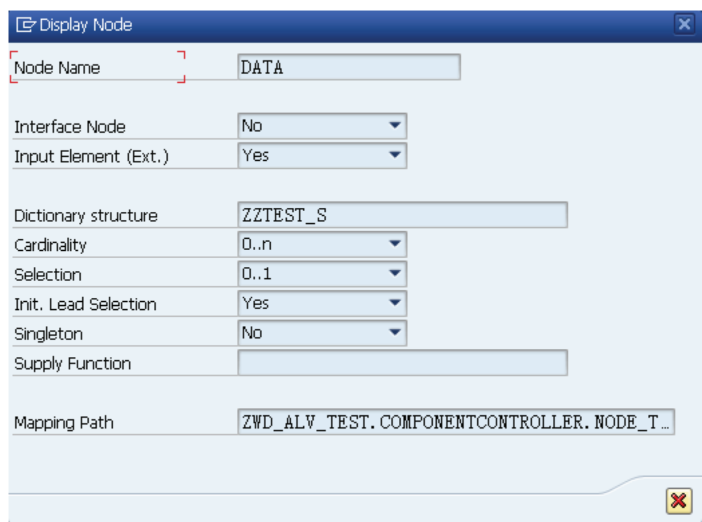


图 6–14

5）选择视图 V_MAIN，选择“Properties”选项卡，添加已声明要使用的组件如图 6–15 所示。

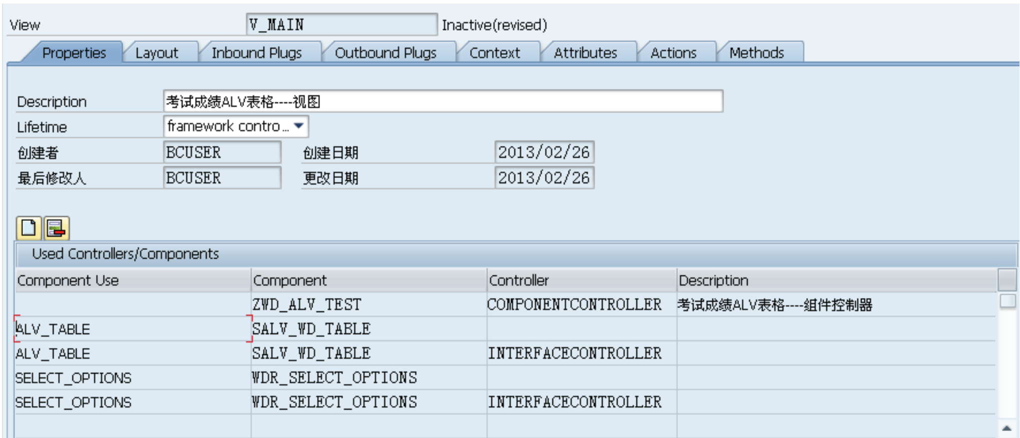


图 6–15

选择“Atrributes”选项卡，添加属性如图 6–16 所示。

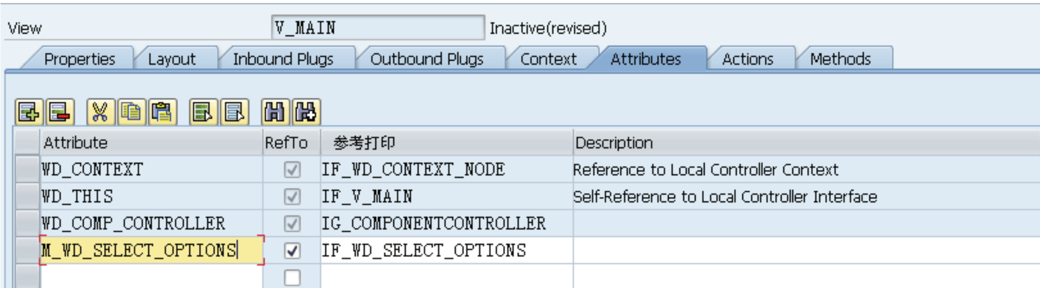


图 6–16

选择“Context”选项卡，创建节点 NODE_TEST 并映射组件控制器中对应的 Context 节点，结果如图 6–17 所示。

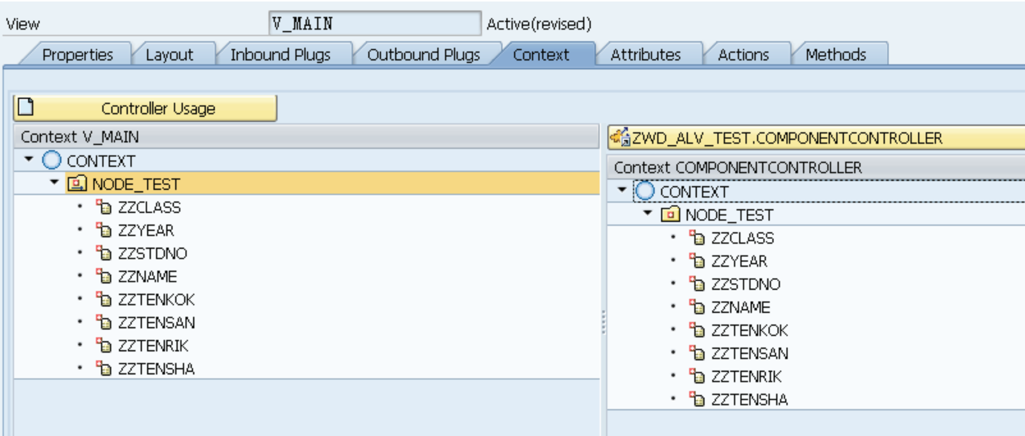


图 6–17

6）选择“Layout”选项卡，添加视图容器 UI 元素和按钮 UI 元素，如图 6-18 所示。布局编辑如图 6-19 所示。



图 6-18　　　　　　　　　　图 6-19

为按钮添加动作，如图 6-20 所示。

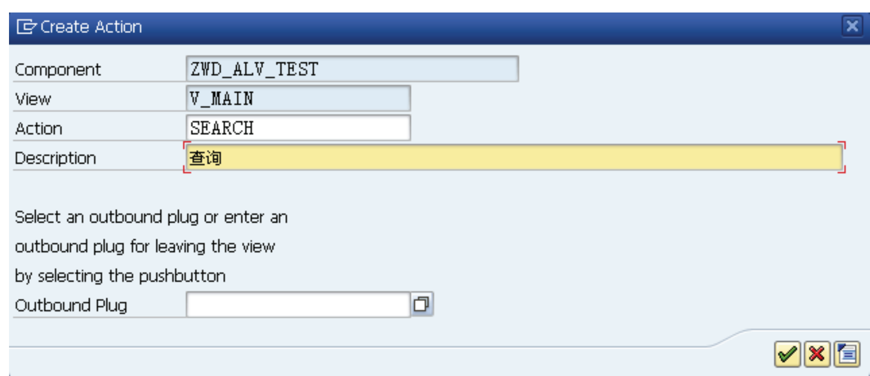


图 6-20

动作处理方法 ONACTIONSEARCH 编辑如下：

```
METHOD ONACTIONSEARCH .
DATA LO_RANGE_ZZCLASS   TYPE REF TO DATA.
FIELD-SYMBOLS: <LFS_ZZCLASS> TYPE TABLE.
DATA LO_RANGE_ZZYEAR   TYPE REF TO DATA.
FIELD-SYMBOLS: <LFS_ZZYEAR> TYPE TABLE.
DATA LV_RC LIKE SY-SUBRC.
DATA LO_ND_NODE_TEST TYPE REF TO IF_WD_CONTEXT_NODE.
DATA LT_NODE_TEST TYPE WD_THIS->ELEMENTS_NODE_TEST.

CALL METHOD WD_THIS->M_WD_SELECT_OPTIONS->GET_RANGE_TABLE_OF_SEL_FIELD
    EXPORTING
      I_ID                  = 'S_ZZCLASS'
    RECEIVING
      RT_RANGE_TABLE = LO_RANGE_ZZCLASS.
ASSIGN LO_RANGE_ZZCLASS->*  TO <LFS_ZZCLASS>.
CALL METHOD WD_THIS->M_WD_SELECT_OPTIONS->GET_RANGE_TABLE_OF_SEL_FIELD
    EXPORTING
      I_ID                  = 'S_ZZYEAR'
```

```
        RECEIVING
            RT_RANGE_TABLE = LO_RANGE_ZZYEAR.
    ASSIGN LO_RANGE_ZZYEAR->*  TO <LFS_ZZYEAR>.

    *  navigate from <CONTEXT> to <NODE_TEST> via lead selection
      LO_ND_NODE_TEST = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->
WDCTX_NODE_TEST ).

    *  @TODO handle non existant child
    *  IF lo_nd_node_test IS INITIAL.
    *  ENDIF.
    *  * @TODO compute values
      SELECT       * FROM  ZZTEST_V
      INTO CORRESPONDING FIELDS OF TABLE LT_NODE_TEST
        WHERE  ZZCLASS  IN <LFS_ZZCLASS>
        AND    ZZYEAR   IN <LFS_ZZYEAR>.
    *
      LO_ND_NODE_TEST->BIND_TABLE( NEW_ITEMS = LT_NODE_TEST SET_INITIAL_
ELEMENTS = ABAP_TRUE ).
    ENDMETHOD.
```

编辑钩子方法 WDDOMODIFYVIEW 如下：

```
    METHOD WDDOMODIFYVIEW .
        DATA LO_COMPONENTINTERFACE TYPE REF TO IF_WD_COMPONENT_USAGE.
        LO_COMPONENTINTERFACE = WD_THIS->WD_CPUSE_SELECT_OPTIONS( ).
        DATA: LT_RANGE           TYPE REF TO DATA.
        DATA LO_INTERFACE TYPE REF TO IWCI_WDR_SELECT_OPTIONS.
        LO_INTERFACE = WD_THIS->WD_CPIFC_SELECT_OPTIONS( ).
        DATA  LO_WD_SELECT_OPTIONS TYPE REF TO IF_WD_SELECT_OPTIONS.
     IF FIRST_TIME = ABAP_TRUE.
       IF   LO_COMPONENTINTERFACE->HAS_ACTIVE_COMPONENT( ) = ABAP_FALSE.
         LO_COMPONENTINTERFACE->CREATE_COMPONENT( ).
       ENDIF.
       CALL METHOD LO_INTERFACE->INIT_SELECTION_SCREEN
         RECEIVING
             R_HELPER_CLASS = LO_WD_SELECT_OPTIONS.
       CALL METHOD LO_WD_SELECT_OPTIONS->SET_GLOBAL_OPTIONS
         EXPORTING
             I_DISPLAY_BTN_CANCEL  = ABAP_FALSE
             I_DISPLAY_BTN_CHECK   = ABAP_FALSE
             I_DISPLAY_BTN_RESET   = ABAP_TRUE
             I_DISPLAY_BTN_EXECUTE = ABAP_FALSE.
       LT_RANGE =
         LO_WD_SELECT_OPTIONS->CREATE_RANGE_TABLE( I_TYPENAME = 'ZZCLASS_E' ).
```

```
          CALL METHOD LO_WD_SELECT_OPTIONS->ADD_SELECTION_FIELD
            EXPORTING
              I_ID          = 'S_ZZCLASS'
              IT_RESULT = LT_RANGE.
          CALL METHOD LO_WD_SELECT_OPTIONS->SET_GLOBAL_OPTIONS
            EXPORTING
              I_DISPLAY_BTN_CANCEL  = ABAP_FALSE
              I_DISPLAY_BTN_CHECK   = ABAP_FALSE
              I_DISPLAY_BTN_RESET   = ABAP_TRUE
              I_DISPLAY_BTN_EXECUTE = ABAP_FALSE.
          LT_RANGE =
            LO_WD_SELECT_OPTIONS->CREATE_RANGE_TABLE( I_TYPENAME = 'ZZYEAR_E' ).
          CALL METHOD LO_WD_SELECT_OPTIONS->ADD_SELECTION_FIELD
            EXPORTING
              I_ID      = 'S_ZZYEAR'
              IT_RESULT = LT_RANGE.

              WD_THIS->M_WD_SELECT_OPTIONS =  LO_WD_SELECT_OPTIONS.
        ENDIF.
      ENDMETHOD.
```

7）选择窗体 W_ALV_TEST，选择“Window”选项卡，在视图容器 UI 元素中添加使用组件中的视图，如图 6-21 所示。

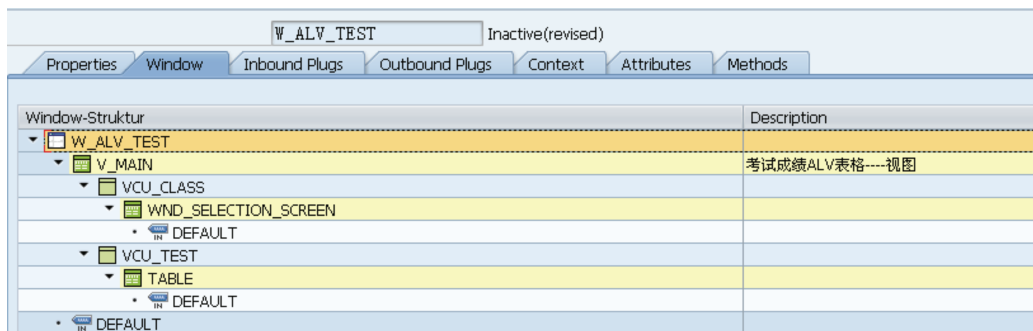


图 6-21

8）为组件创建 Web Dynpro Apllication，如图 6-22 所示。

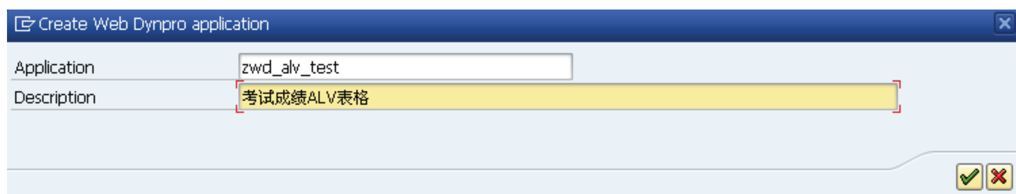


图 6-22

9）运行 Web Dynpro 应用程序，测试显示结果，如图 6-23 所示。

班级信息

重设

班级: ◇ 1A 至

学年: ◇ 至

查询

成绩列表

View [标准视图] Print Version Export Filter Settings

班级	学年	学生学号	姓名	语文成绩	数学成绩	自然成绩	思品成绩
1A				0	0	0	0
1A	2009	00001	郑喜定	60	68	70	81
1A	2009	00002	魏喜奎	62	65	74	83
1A	2009	00003	金英	62	65	71	89
1A	2009	00004	洪喜官	61	63	75	88
1A	2009	00005	朴珍	64	66	75	88
1A	2009	00006	王喜凤	64	66	75	88
1A	2009	00007	刘喜仁	64	66	76	88
1A	2009	00008	白喜医	62	66	72	88
1A	2009	00009	赵喜庆	62	63	72	82

图 6-23

单击 Print Version 按钮，可生成 PDF 文件，如图 6-24 所示，

alv_export[1].pdf - Foxit Reader 2.2 - [alv_export[1].pdf]

File Edit View Language Document Tools Advanced Window Help

学生学号	姓名	语文成绩	数学成绩	自然成绩	思品成绩	班级	学年
		0	0	0	0	1A	0000
00001	郑喜定	60	68	70	81	1A	2009
00002	魏喜奎	62	65	74	83	1A	2009
00003	金英	62	65	71	89	1A	2009
00004	洪喜官	61	63	75	88	1A	2009
00005	朴珍	64	66	75	88	1A	2009
00006	王喜凤	64	66	75	88	1A	2009
00007	刘喜仁	64	66	76	88	1A	2009
00008	白喜医	62	66	72	88	1A	2009
00009	赵喜庆	62	63	72	82	1A	2009
00010	张喜早	62	63	74	85	1A	2009

图 6-24

注意：这里需要配置 RFC，如图 6-25 所示。

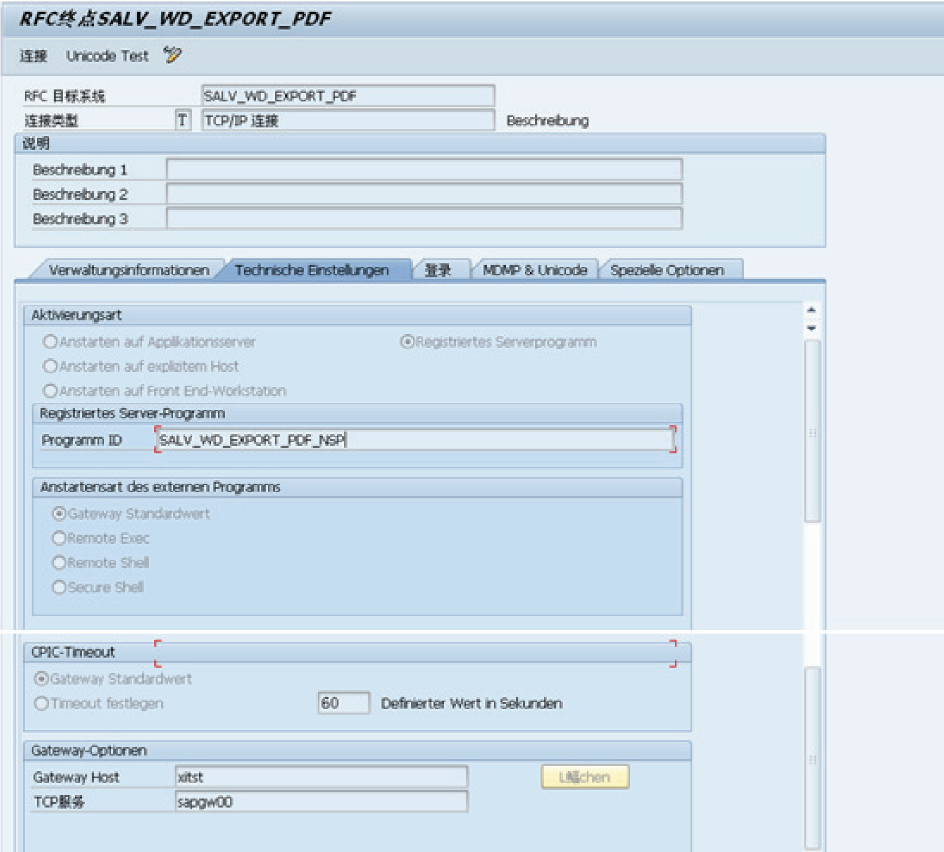


图 6-25

在网关服务器中，添加对应服务，如图 6-26 所示。

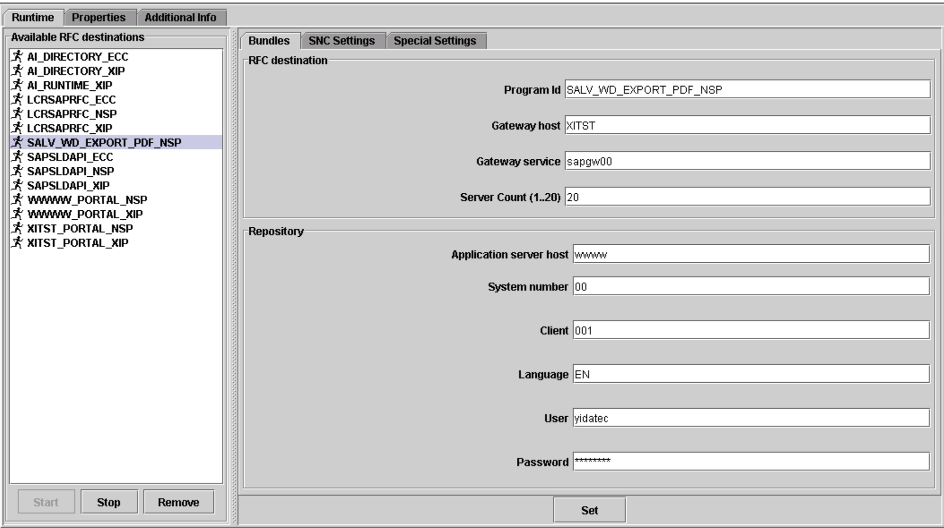


图 6-26

单击 Export 按钮，按照图 6-27 所示的路径下载文件到 Excel 文件。

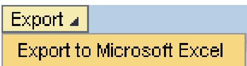


图 6-27

下载文件结果如图 6-28 所示。

班级	学年	学生学号	姓名	语文成绩	数学成绩	自然成绩	思品成绩
1A	0000			0	0	0	0
1A	2009	00001	郑喜定	60	68	70	81
1A	2009	00002	魏喜奎	62	65	74	83
1A	2009	00003	金英	62	65	71	89
1A	2009	00004	洪喜官	61	63	75	88
1A	2009	00005	朴珍	64	66	75	88
1A	2009	00006	王喜凤	64	66	75	88
1A	2009	00007	刘喜仁	64	66	76	88
1A	2009	00008	白喜医	62	66	72	88
1A	2009	00009	赵喜庆	62	63	72	82
1A	2009	00010	张喜早	62	63	74	85

图 6-28

注：集成 ADOBE FLASH ISLANDS、文件操作、门户集成（PORTAL INTEGRATION）、集成表单（INTEGRATING FORMS）、ALV 在 WEB DYNPRO ABAP 的应用均属于 Web Dynpro For ABAP 集成技术。就集成技术而言，除了以上几种，Web Dynpro For ABAP 还可以与 Office 集成、与 BI 集成、与 CAF 引导程序集成，在视图设计器中提供了相关的 UI 元素类别整合的功能。

第 7 章　高 级 概 念

7.1　组件辅助类

对于每一个 Web Dynpro 组件，都可以为其分配一个辅助类 Class。这个类必须继承抽象类 CL_WD_COMPONENT_ASSISTANCE。该组件的辅助类有以下优点：

- 程序员可以编辑组件所需的代码，但是这些代码可以不直接连接到视图 View 的布局或者控制器的某个功能。这样就可以实现应用层的调用或者基于用户接口的数据编辑。
- 从性能角度讲，辅助类的方法调用要好于 Web Dynpro 控制器中方法的调用。
- 第二个比较重要的辅助类功能是用于管理动态的文本文字。这些文本文字在辅助类的文本池中作为文本符号（Text Symbols）存储，只有在运行时被读取。

辅助类在组件被调用时自动被实例化。该实例通过属性 WD_ASSIST 对于组件中各个控制器都是有效的。

类 CL_WD_COMPONENT_ASSISTANCE 提供中心函数 GET_TEXT，通过它，Web Dynpro 组件可以访问辅助类的文本符号。使用实例属性 WD_ASSIST 和方法 WD_COMPONENT_ASSISTANCE~GET_TEXT( )，程序员可以在组件的各个控制器读取到辅助类的文本符号。当方法被调用时，文本符号的 3 个字节的数字 ID 通过 KEY 参数传递到方法中。

```
method MY_CONTROLLER_METHOD .
data: my_text type string.
my_text = WD_ASSIST->IF_WD_COMPONENT_ASSISTANCE~GET_TEXT( KEY = ‘001’ ).

endmethod.
```

下面介绍编辑 Web Dynpro 应用程序的方法。

通过自定义的组件辅助类完成取值赋值操作的步骤如下。

1）在 ABAP 工作台中的编辑界面创建 Web Dynpro 组件。

选择“Web Dynpro Comp./Intf.”，在相应的文本框中输入程序的名称，如图 7-1 所示。

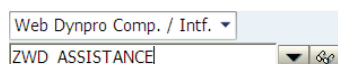


图 7-1

单击图 7-1 中的按钮创建组件，如图 7-2 所示。

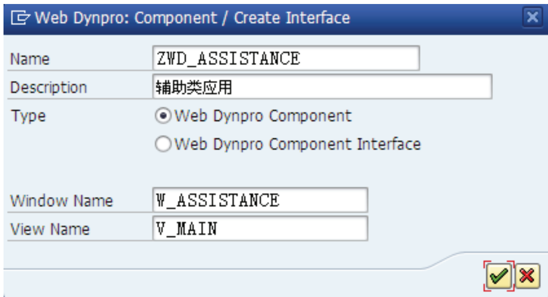


图 7-2

编辑结果如图 7-3 所示。

Object Name	Description
ZWD_ASSISTANCE	辅助类应用
COMPONENTCONTROLLER	辅助类应用---组件控制器
Component Interface	
INTERFACECONTROLLER	辅助类应用---组件控制器
Interface Views	
W_ASSISTANCE	辅助类应用---窗体
Views	
V_MAIN	辅助类应用---视图
Windows	
W_ASSISTANCE	辅助类应用---窗体

图 7-3

2）在 ABAP 工作台中的编辑界面创建组件辅助类。

在右侧树形结构中双击组件 ZWD_ASSISTANCE ，添加组件辅助类，如图 7-4 所示。

Web Dynpro Component	ZWD_ASSISTANCE	Inactive	
Description	辅助类应用		
Assistance Class	ZCL_ASSISTANCE		
Created By	BCUSER	Created On	2011/10/20
Last Changed By	BCUSER	Changed On	2011/10/20
Original Lang.	ZH	Package	ZWD_TRAINING
Accessibility Checks Active			

图 7-4

按〈Enter〉键创建 ZCL_ASSISTANCE，双击进行编辑，选择“Type”选项卡，定义类型如图 7-5 所示。

Klassenschnittstelle ZCL_ASSISTANCE 已实现 /

Eigenschaften | Interfaces | Friends | Attribute | Methoden | Ereignisse | Typen | 别名

筛选器

类型	可见性	Typisi...	参考打印		说明
PARAMS	Public	Type			Text参数
PARAMS_TABLE	Public	Type			Text参数表
		Type			

图 7-5

单击按钮，自定义参数类型如下：

```
PUBLIC SECTION.
    TYPES:
      BEGIN OF PARAMS,
          PARA1 TYPE STRING,
          PARA2 TYPE STRING,
          PARA3 TYPE STRING,
          PARA4 TYPE STRING,
        END OF PARAMS .
    TYPES:
      PARAMS_TABLE TYPE TABLE OF PARAMS .
```

单击标准工具栏中的按钮，选择“Methoden”选项卡，添加自定义方法如图 7-6 所示。

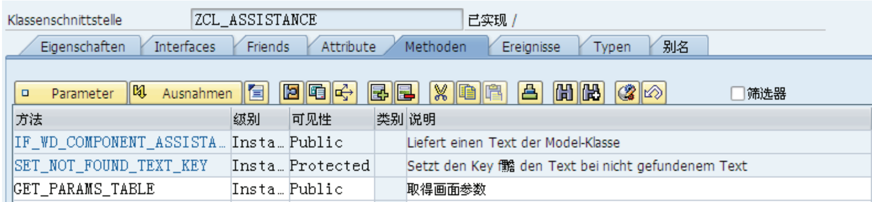


图 7-6

选中方法 GET_PARAMS_TABLE，单击 Parameter 按钮，定义参数如图 7-7 所示。

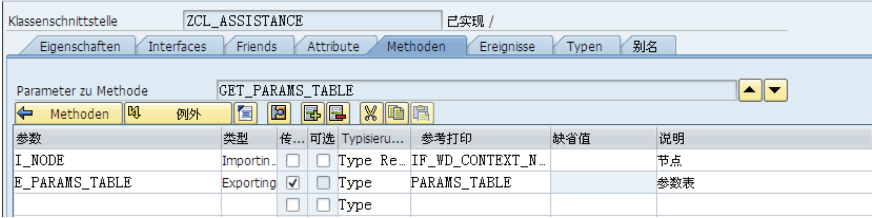


图 7-7

方法编辑如下：

```
METHOD GET_PARAMS_TABLE.
  DATA LO_EL_NODE_INPUT TYPE REF TO IF_WD_CONTEXT_ELEMENT.
  DATA LS_PARAMS TYPE PARAMS.

  LO_EL_NODE_INPUT = I_NODE->GET_ELEMENT( ).
  LO_EL_NODE_INPUT->GET_STATIC_ATTRIBUTES(
    IMPORTING
      STATIC_ATTRIBUTES = LS_PARAMS ).

  APPEND LS_PARAMS TO E_PARAMS_TABLE.
ENDMETHOD.
```

单击标准工具栏中的按钮，选择“Attribute”选项卡，单击按钮，添加自定义属

性 CONSTANT_PARAMS 如下：

```
CONSTANTS:    BEGIN OF CONSTANT_PARAMS,
                PARA1 TYPE STRING VALUE '&PARA1&',
                PARA2 TYPE STRING VALUE '&PARA2&',
                PARA3 TYPE STRING VALUE '&PARA3&',
                PARA4 TYPE STRING VALUE '&PARA4&',
              END OF CONSTANT_PARAMS .
```

编辑结果如图 7-8 所示。

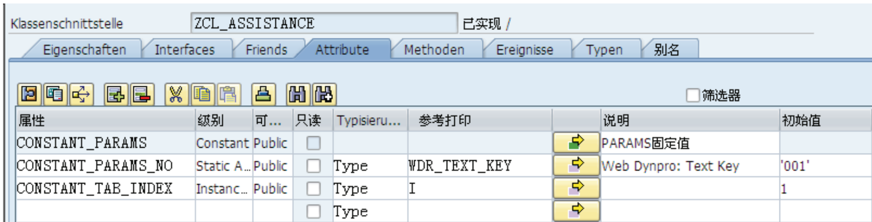
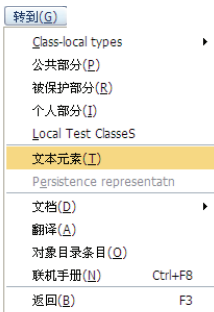


图 7-8

按照图 7-9 所示的路径定义文本元素。

编辑结果如图 7-10 所示。



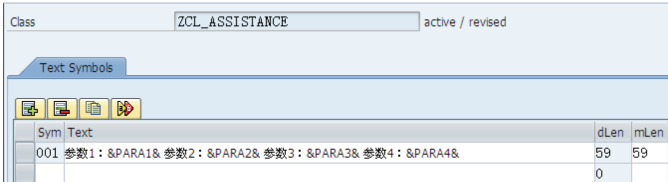


图 7-9　　　　　　　　　　　　　图 7-10

3）在 ABAP 工作台中的编辑界面编辑组件。

单击标准工具栏中的按钮，选择组件 ZWD_ASSISTANCE 视图 V_MAIN 的“Context”选项卡，添加 Context 节点如图 7-11 所示。

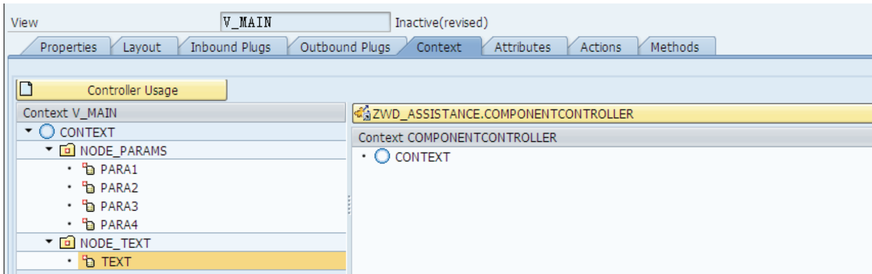


图 7-11

选择“Layout”选项卡，编辑 UI 元素如图 7-12 所示。

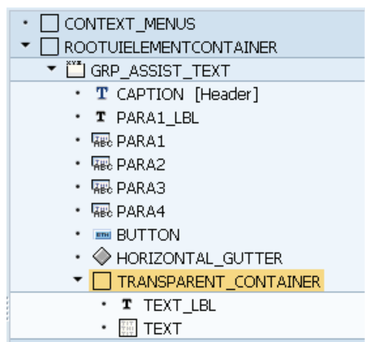


图 7-12

编辑结果如图 7-13 所示。

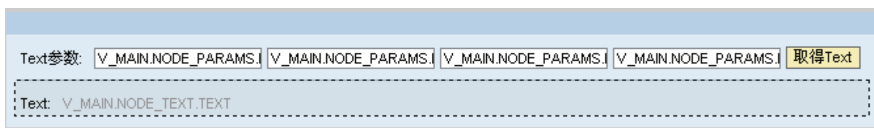


图 7-13

为按钮 UI 元素添加动作如图 7-14 所示。

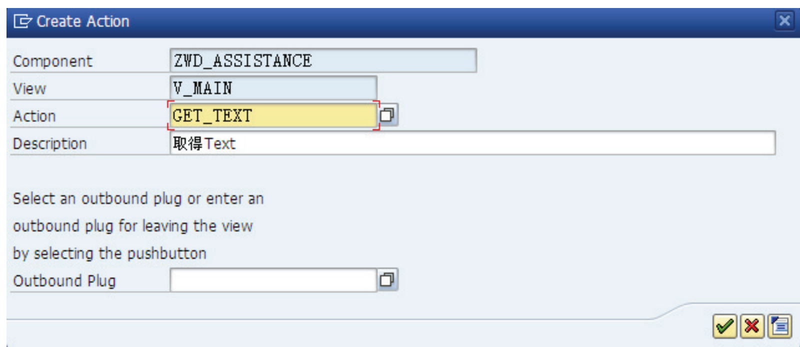


图 7-14

动作对应处理程序编辑如下：

```
        METHOD ONACTIONGET_TEXT .
        * 取得参数并编辑内表
            DATA LO_ND_NODE_PARAMS TYPE REF TO IF_WD_CONTEXT_NODE.
            DATA LS_NODE_PARAMS TYPE WD_THIS->ELEMENT_NODE_PARAMS.
            DATA LT_NODE_PARAMS TYPE WD_THIS->ELEMENTS_NODE_PARAMS.

        *   navigate from <CONTEXT> to <NODE_PARAMS> via lead selection
            LO_ND_NODE_PARAMS = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->WD
CTX_NODE_PARAMS ).
```

```
    WD_ASSIST->GET_PARAMS_TABLE(
        EXPORTING
            I_NODE            = LO_ND_NODE_PARAMS
        IMPORTING
            E_PARAMS_TABLE = LT_NODE_PARAMS
                    ).
    READ TABLE LT_NODE_PARAMS INTO LS_NODE_PARAMS INDEX 1.
* 根据参数取得完整 Text
    DATA LO_ND_NODE_TEXT TYPE REF TO IF_WD_CONTEXT_NODE.
    DATA LO_EL_NODE_TEXT TYPE REF TO IF_WD_CONTEXT_ELEMENT.
    DATA LS_NODE_TEXT TYPE WD_THIS->ELEMENT_NODE_TEXT.

* navigate from <CONTEXT> to <NODE_TEXT> via lead selection
        LO_ND_NODE_TEXT = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->WD
CTX_NODE_TEXT ).
* get element via lead selection
    LO_EL_NODE_TEXT = LO_ND_NODE_TEXT->GET_ELEMENT( ).

* @TODO fill static attributes
    LS_NODE_TEXT-TEXT = WD_ASSIST->IF_WD_COMPONENT_ASSISTANCE~GET_TEXT(
              KEY    = zcl_assistance=>CONSTANT_PARAMS_NO
              PARA1  = LS_NODE_PARAMS-PARA1
              PARA2  = LS_NODE_PARAMS-PARA2
              PARA3  = LS_NODE_PARAMS-PARA3
              PARA4  = LS_NODE_PARAMS-PARA4
                    ).

* set all declared attributes
    LO_EL_NODE_TEXT->SET_STATIC_ATTRIBUTES(
        STATIC_ATTRIBUTES = LS_NODE_TEXT ).
ENDMETHOD.
```

转到 Methods 标签页，编辑钩子方法 WDDOINIT 如下：

```
METHOD WDDOINIT .
* 为画面赋初始值
    DATA LO_ND_NODE_PARAMS TYPE REF TO IF_WD_CONTEXT_NODE.
    DATA LO_EL_NODE_PARAMS TYPE REF TO IF_WD_CONTEXT_ELEMENT.
    DATA LS_NODE_PARAMS TYPE WD_THIS->ELEMENT_NODE_PARAMS.
* navigate from <CONTEXT> to <NODE_PARAMS> via lead selection
    LO_ND_NODE_PARAMS = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->WD
CTX_NODE_PARAMS ).
* get element via lead selection
    LO_EL_NODE_PARAMS = LO_ND_NODE_PARAMS->GET_ELEMENT( ).
* @TODO fill static attributes
    LS_NODE_PARAMS = WD_ASSIST->CONSTANT_PARAMS.
* set all declared attributes
```

```
    LO_EL_NODE_PARAMS->SET_STATIC_ATTRIBUTES(
        STATIC_ATTRIBUTES = LS_NODE_PARAMS ).
* 根据参数取得完整 Text
    DATA LO_ND_NODE_TEXT TYPE REF TO IF_WD_CONTEXT_NODE.
    DATA LO_EL_NODE_TEXT TYPE REF TO IF_WD_CONTEXT_ELEMENT.
    DATA LS_NODE_TEXT TYPE WD_THIS->ELEMENT_NODE_TEXT.

*  navigate from <CONTEXT> to <NODE_TEXT> via lead selection
    LO_ND_NODE_TEXT = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->WDCT
X_NODE_TEXT ).
*  get element via lead selection
    LO_EL_NODE_TEXT = LO_ND_NODE_TEXT->GET_ELEMENT( ).
*  @TODO fill static attributes
    LS_NODE_TEXT-TEXT = WD_ASSIST->IF_WD_COMPONENT_ASSISTANCE~GET_TEXT(
                KEY     = zcl_assistance=>CONSTANT_PARAMS_NO
                PARA1  = LS_NODE_PARAMS-PARA1
                PARA2  = LS_NODE_PARAMS-PARA2
                PARA3  = LS_NODE_PARAMS-PARA3
                PARA4  = LS_NODE_PARAMS-PARA4
                  ).
*  set all declared attributes
    LO_EL_NODE_TEXT->SET_STATIC_ATTRIBUTES(
        STATIC_ATTRIBUTES = LS_NODE_TEXT ).

ENDMETHOD.
```

4）在 ABAP 工作台中的编辑界面编辑 Web Dynpro Application，如图 7-15 所示。

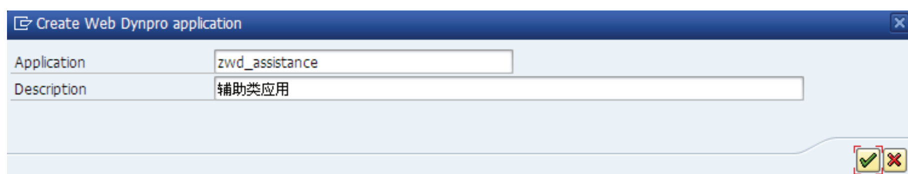


图 7-15

5）在 ABAP 工作台运行 Web Dynpro Application，如图 7-16 所示。

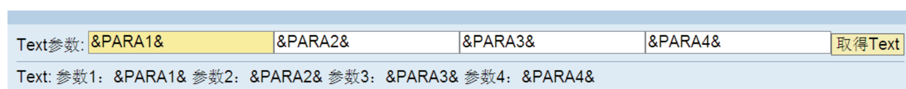


图 7-16

7.2 Web Dynpro 应用中的服务调用（Service Call）

通过服务调用功能（Service Call），使得 Web Dynpro 组件内调用已存在的函数或类中的

方法。

（1）前提（Prerequisites）

使用服务调用将现有的功能函数模块包含到当前的 Web Dynpro 应用程序中，可以访问在当前系统中可用的函数。

（2）特点（Features）

为了简化创建过程，在编辑环境中可以使用向导。在该向导的帮助下，在所选择的控制器中自动生成所需的所有 Context 元素。此外，该向导会在调用功能函数的控制器中自动创建一个方法并确定传递的参数。

7.2.1 创建一个 Service Call

要创建一个服务调用，可以使用在 ABAP Workbench 中的 Web Dynpro 工具内的一个易于使用的向导。

（1）启动向导

要启动该向导，在工作台窗口的左侧将光标定位在对象列表中要编辑的 Web Dynpro 组件（ZWD_STUDENT）。单击鼠标右键，打开 Context 菜单，按照图 7-17 所示的路径启动向导。

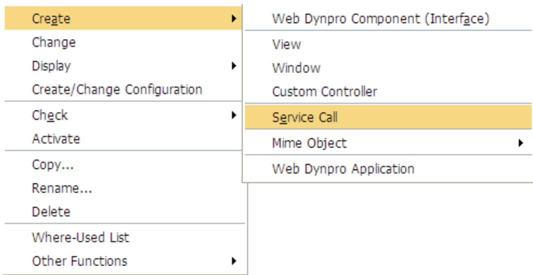


图 7-17

（2）选择控制器

在向导的第二个对话框中，程序员可以选择是否将服务调用嵌入到现有的控制器，或创建一个新的控制器用于此目的，如图 7-18 所示。

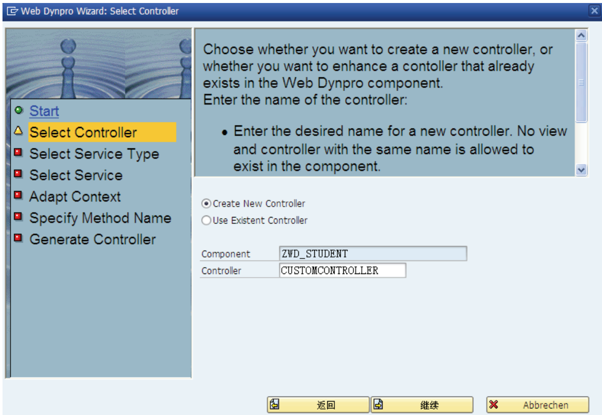


图 7-18

注：服务调用只能嵌入到全局控制器中，即在组件控制器中或自定义的定制控制器中。

在这一点上，一般选择创建一个新的自定义控制器用于保存数据。由于是全局控制器，因此它可以被其他控制器引用。

单击继续按钮，如图 7-19 所示。

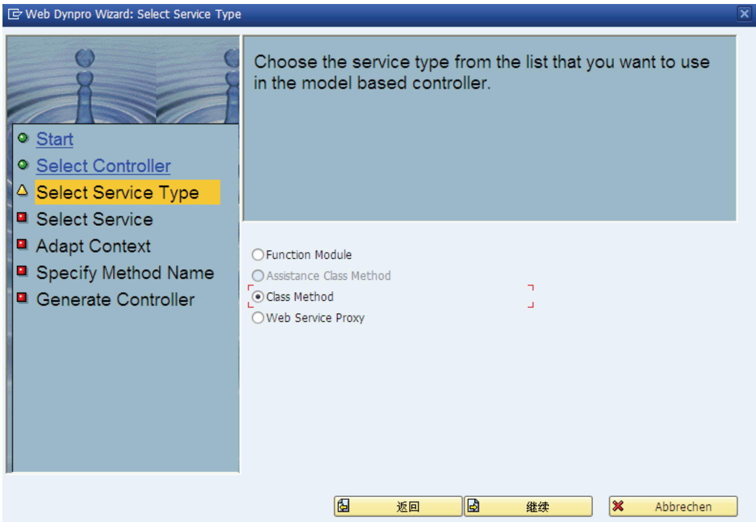


图 7-19

在图 7-19 所示的对话框中可以选择 Function Module、Class Method 及 Web Service Proxy。这里选中“Class Method”单选按钮，单击继续按钮，显示的向导页面如图 7-20 所示。

（3）必要的方法和 Context 元素

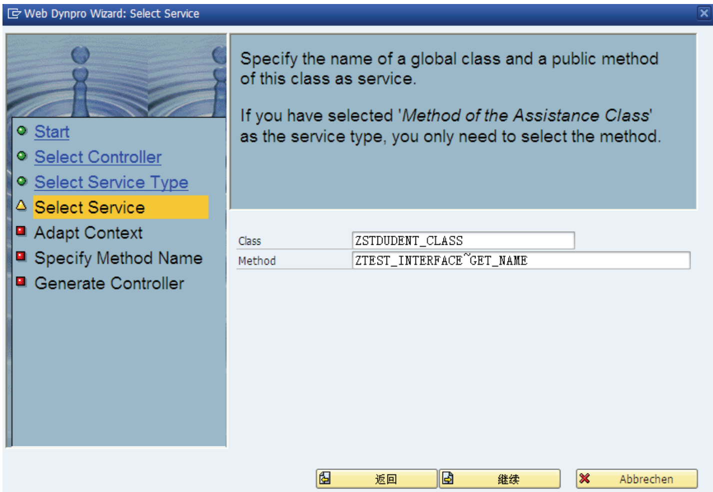


图 7-20

这里选中要调用的方法，单击继续按钮，显示的向导页面如图 7-21 所示。

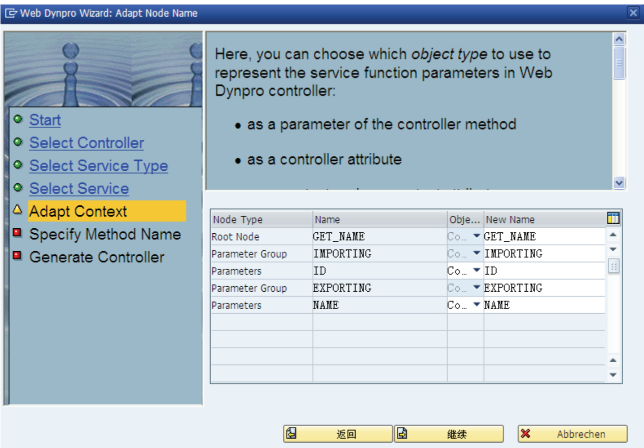


图 7-21

定义所需 Context 节点，单击继续按钮，显示的向导页面如图 7-22 所示。

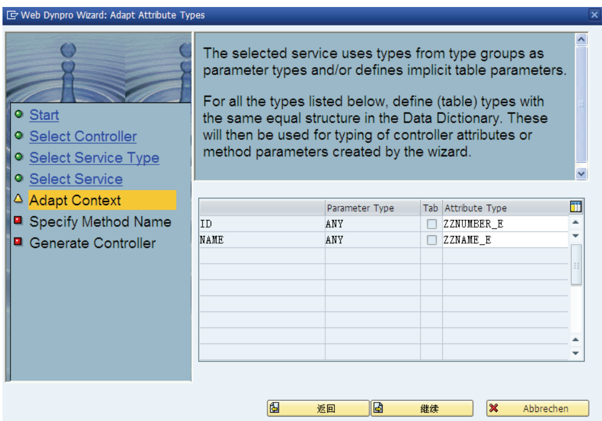


图 7-22

定义 Context 节点属性类型，单击继续按钮，显示的向导页面如图 7-23 所示。

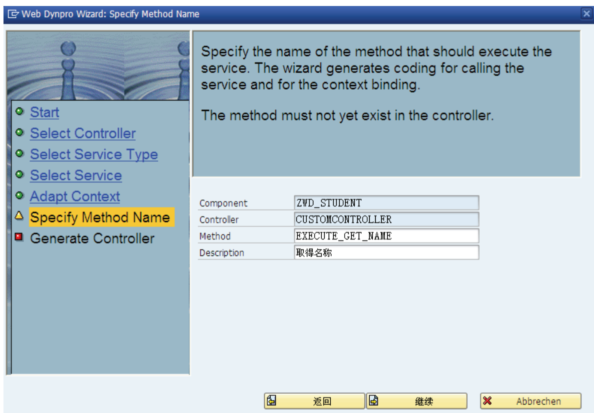


图 7-23

定义自定义控制器所需的方法，单击继续按钮，显示的向导页面如图 7-24 所示。

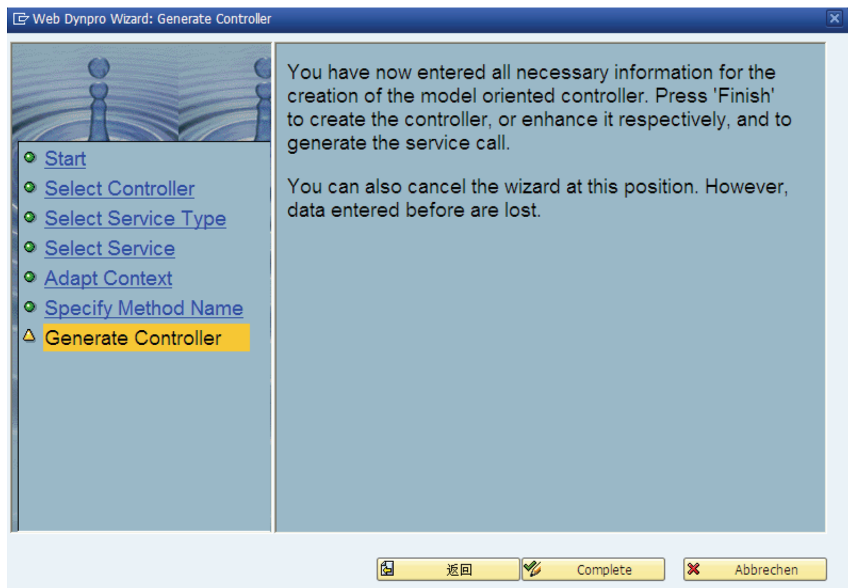


图 7-24

单击Complete按钮，完成配置。

（4）完成选项

当确认最后一个对话框时，编码生成器被触发。之后就可以使用所配置的 Web Dynpro 组件所需的方法和 Context 属性。

配置结果如图 7-25 所示。

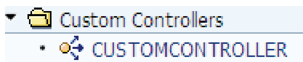


图 7-25

配置时预定义 Context 生成结果如图 7-26 所示。

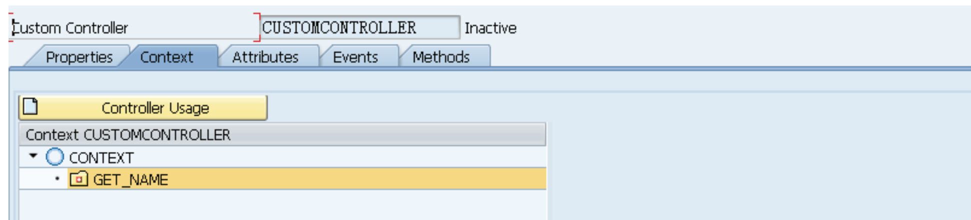


图 7-26

配置时预定义属性生成结果如图 7-27 所示。

Custom Controller CUSTOMCONTROLLER Inactive

Properties | Context | Attributes | Events | Methods

Attribute	Public	RefTo	参考打印	Description
WD_CONTEXT	☐	☑	IF_WD_CONTEXT_NODE	Reference to Local Controller Context
WD_THIS	☐	☑	IF_CUSTOMCONTROLLER	Self-Reference to Local Controller Interface
ID	☐	☐	ZZNUMBER_E	番号
M_ZSTDUDENT_CLASS	☐	☑	ZSTDUDENT_CLASS	
NAME	☐	☐	ZZNAME_E	名称

图 7-27

配置时预定义方法生成结果如图 7-28 所示。

Custom Controller CUSTOMCONTROLLER Inactive

Properties | Context | Attributes | Events | Methods

Method	Method Type	Description	Event	Controller
EXECUTE_GET_NAME	0 Method	取得名称		
WDDOEXIT	0 Method	Cleanup Method of Controller		
WDDOINIT	0 Method	Initialization Method of Controller		
	0 Method			

图 7-28

编辑方法 EXECUTE_GET_NAME 如下：

```
METHOD EXECUTE_GET_NAME .
    IF NOT WD_THIS->M_ZSTDUDENT_CLASS IS BOUND.
      CREATE OBJECT WD_THIS->M_ZSTDUDENT_CLASS.
    ENDIF.
* declarations for context navigation

* declarations for parameters

* get all involved child nodes

* get input from context

* the invocation - errors are always fatal !!!
    WD_THIS->M_ZSTDUDENT_CLASS->ZTEST_INTERFACE~GET_NAME(
        EXPORTING
            ID =                    WD_THIS->ID
        IMPORTING
            NAME =                  WD_THIS->NAME
 ).
* store output to context

ENDMETHOD.
```

7.2.2 使用 Service Call

（1）前提（Prerequisites）

已经在已知的组件中定义服务调用。

（2）操作

服务调用的使用与控制器中手动创建的方法的使用有所不同。除了由向导的程序自动创建 Context 节点和属性外，生成的方法可以跨控制器使用。

（3）示例

用户若想从数据库表中读取数据并在现有的功能模块的帮助下在视图上显示它，首先要在 Web Dynpro 组件中创建一个功能模块的服务调用，然后可以看到在选择作为全局控制器的 Context 视图中已经创建了所需的 Context 节点和属性以及调用功能模块的方法，如图 7-29 所示。

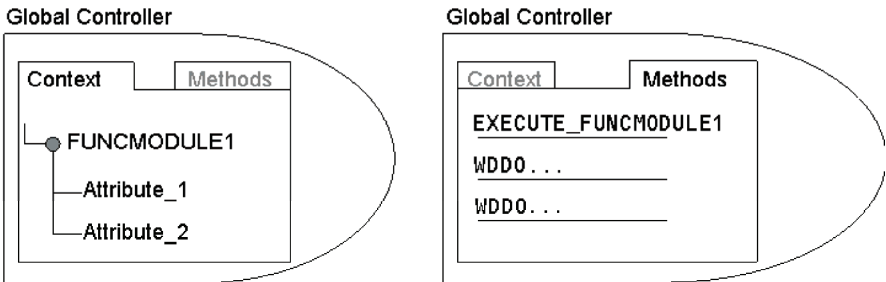


图 7-29

方法 EXECUTE_FUNCMODULE1 已经自动生成。

此时，功能模块对于组件是可用的。现在需要选择一个视图，以便在浏览器中显示数据库表中的数据。接着使用选定的视图控制器以全局控制器（组件控制器）为输入。随后，映射节点 FUNCMODUL1 到视图控制器 Context 中，有相同的名称节点在视图控制器中自动生成，如图 7-30 所示。

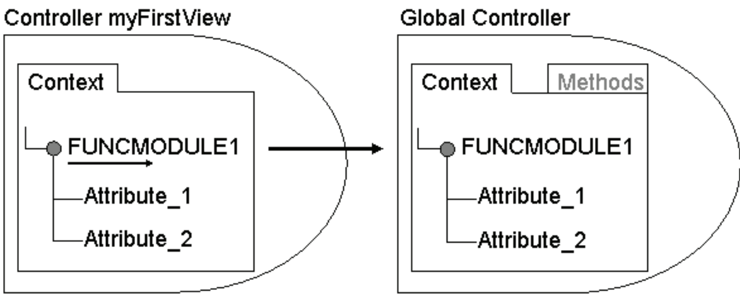


图 7-30

为了将数据库表中的数据绑定到视图控制器 Context 中 Context 节点的 FUNCMODUL1 上，作为全局控制器的供给函数的方法 EXECUTE_FUNCMODULE1 被调用。要做到这一点，一个供给函数的方法 EXECUTE_FUNCMODULE1 在全局控制器中被创建和修改，如图 7-31 所示。

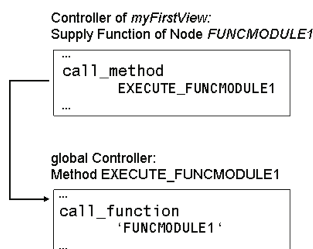


图 7-31

1）在 ABAP 工作台中的自定义控制器编辑界面，完善自定义控制器的方法。

此例中，自定义控制器中的 Context 用于接收数据，在此引用组件控制器，选择自定义控制器，选择“Properties”选项卡，设定组件使用如图 7-32 所示。

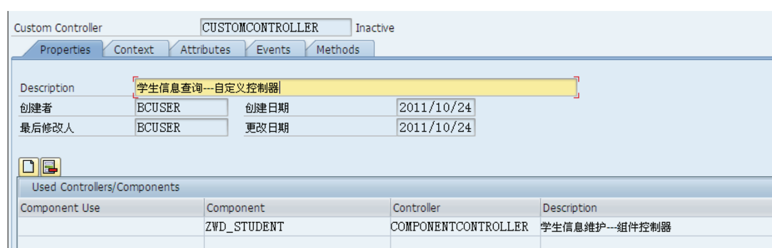


图 7-32

选择“Context”选项卡，将节点 NODE_STUDENT 拖至 GET_NAME 上，为 GET_NAME 创建属性并映射，如图 7-33 所示。

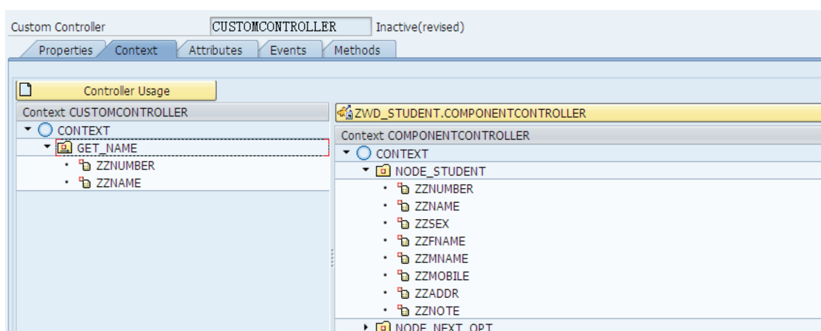


图 7-33

注：该处为部分映射。

选择“Methods”选项卡，编辑方法 EXECUTE_GET_NAME 如下：

```
METHOD EXECUTE_GET_NAME .

  IF NOT WD_THIS->M_ZSTDUDENT_CLASS IS BOUND.
    CREATE OBJECT WD_THIS->M_ZSTDUDENT_CLASS.
  ENDIF.
```

```
    * declarations for context navigation
      DATA LO_ND_GET_NAME TYPE REF TO IF_WD_CONTEXT_NODE.

      DATA LO_EL_GET_NAME TYPE REF TO IF_WD_CONTEXT_ELEMENT.
      DATA LS_GET_NAME TYPE WD_THIS->ELEMENT_GET_NAME.
    * navigate from <CONTEXT> to <GET_NAME> via lead selection
      LO_ND_GET_NAME   =   WD_CONTEXT->GET_CHILD_NODE(   NAME   =   WD_THIS->
WDCTX_GET_NAME ).
    * get element via lead selection
      LO_EL_GET_NAME = LO_ND_GET_NAME->GET_ELEMENT( ).
    * get all declared attributes
      LO_EL_GET_NAME->GET_STATIC_ATTRIBUTES(
        IMPORTING
          STATIC_ATTRIBUTES = LS_GET_NAME ).
    * declarations for parameters
      WD_THIS->ID = LS_GET_NAME-ZZNUMBER.
    * the invocation - errors are always fatal !!!
      WD_THIS->M_ZSTDUDENT_CLASS->ZTEST_INTERFACE~GET_NAME(
        EXPORTING
          ID =                              WD_THIS->ID
        IMPORTING
          NAME =                            WD_THIS->NAME
      ).

    * store output to context
      LS_GET_NAME-ZZNAME   = WD_THIS->NAME.
      LO_EL_GET_NAME->SET_STATIC_ATTRIBUTES(
        EXPORTING
          STATIC_ATTRIBUTES = LS_GET_NAME ).
    ENDMETHOD.
```

2）在 ABAP 工作台中的视图编辑界面，在视图中调用自定义控制器的方法。

选择视图 V_STUDENT_DEL，选择“Properties”选项卡，设定组件使用如图 7-34 所示。

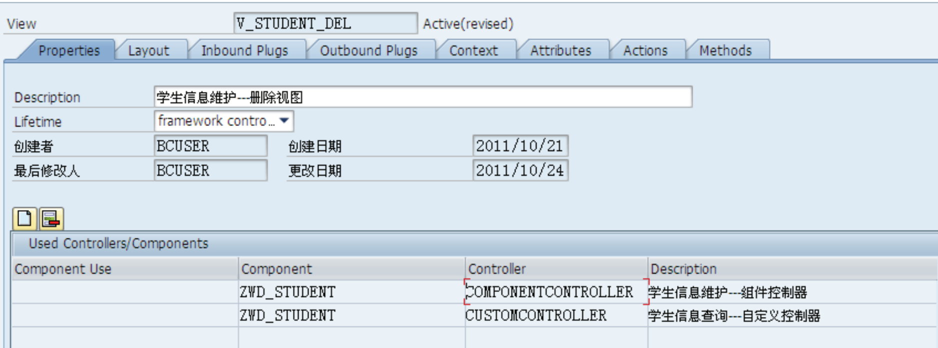


图 7-34

选择“methods”选项卡，编辑钩子方法 WDDOMODIFYVIEW 如下：

```
METHOD WDDOMODIFYVIEW .
  DATA LO_CUSTOMCONTROLLER TYPE REF TO IG_CUSTOMCONTROLLER .
  LO_CUSTOMCONTROLLER =     WD_THIS->GET_CUSTOMCONTROLLER_CTR( ).
  LO_CUSTOMCONTROLLER->EXECUTE_GET_NAME(    ).
ENDMETHOD.
```

选择视图 V_STUDENT_UPD，按照同样的步骤完善组件功能操作。

3）在 ABAP 工作台中的 Web Dynpro Application 编辑界面，测试在视图中调用自定义控制器的方法。

运行 Web Dynpro，应用 zwd_student_del，如图 7-35 所示。

学生信息
键值信息
学生学号: 00002
姓名:
确定

图 7-35

输入学生学号，并单击确定按钮，如图 7-36 所示。

学生信息
键值信息
学生学号: 00002
姓名: 魏喜奎
主体信息
性别: 男性
父亲: 魏东亭
母亲: 张娜
联系电话: 139876×××××
家庭住址: 北京鲁迅路129号
备注: 修改后
保存

图 7-36

运行 Web Dynpro，应用 zwd_student_upd，如图 7-37 所示。

学生信息
键值信息
学生学号: 00003
姓名:
确定

图 7-37

输入学生学号，并单击确定按钮，如图 7-38 所示。

图 7-38

7.3 使用对话框

对话框在 Web Dynpro 视图上，用于显示具体的信息或可能的设置。对话框退出后，对话框下的视图被激活，或跳转到另一个试图。有以下两种不同类型的对话框：

1）模态对话框（Modal）。在当前浏览器窗口中打开一个模态对话框。

注意：每一个模态对话框都有自己的阶段模型实例。

2）外部对话框。外部对话框在一个单独的浏览器窗口中打开，并且独立于原来的窗口，可以在屏幕上移动。

在 Web Dynpro 应用程序中，对话框都是通过一个额外的窗口实现的，并且一般都在一个动作的事件处理程序中被调用（如有必要，在阶段模型的所有其他方法中都可以使用）。组件控制器包含接口 IF_WD_WINDOW_MANAGER，使用它可以创建并打开一个新窗口的对话框 （在创建过程中，相应的组件控制器的使用为每个视图控制器自动设置）。在大多数情况下，在应用程序中使用一个模态对话框。

注：不再使用的参数 MODAL。

集成的控件 UI 元素不支持 Web Dynpro ABAP 对话框。集成的所有的 UI 元素继承 AbstractActiveComponent 组件。

- Gantt（甘特图）。
- Network（网络）。
- OfficeControl（办公控制）。
- InteractiveForm（交互报表）。

7.3.1 同一组件中调用对话框

如果用户想显示对话框，则相关内容要连接到当前组件，以此为目的，程序员应该在这个组件中创建相应的窗体。

接口 IF_WD_WINDOW_MANAGER 的 CREATE_WINDOW 方法允许程序员在事件处理程序方法中，在运行时显示的窗体上创建一个对话框。打开的对话框窗体的名称作为参数传递给 CREATE_WINDOW 方法。属性类型 IF_WD_WINDOW 必须在视图控制器中声明作为对话框。在下面的例子中，这是属性 m_popup1_1，如图 7-39 所示。

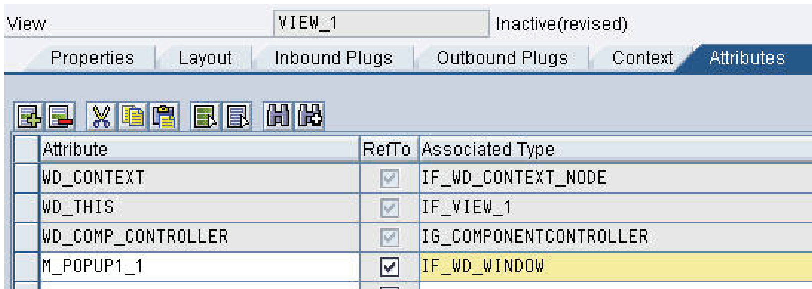


图 7-39

用作该对话框（在这种情况下，POPUP1_1）的窗体必须已经在组件中存在。

```
method onactionpopup1_1 .
  data: l_cmp_api type ref to if_wd_component,
  l_window_manager type ref to if_wd_window_manager.
  l_cmp_api = wd_comp_controller->wd_get_api( ).
  l_window_manager = l_cmp_api->get_window_manager( ).
  if wd_this->m_popup1_1 is initial.
    wd_this->m_popup1_1 = l_window_manager->create_window(
    window_name = 'POPUP1_1'
    button_kind = if_wd_window=>co_buttons_yesnocancel
    message_type = if_wd_window=>co_msg_type_question ).
  endif.
  wd_this->m_popup1_1->open( ).
endmethod.                    "onactionpopup1_1
```

注：CREATE_WINDOW 方法仅创建了新的对话框，它不会打开它。要打开对话框，还需要 OPEN 方法。

（1）对话框按钮

可以使用 BUTTON_KIND 参数指定哪些按钮出现在对话框中。在上面的例子中，设置的 CO_BUTTONS_YESNOCANCEL 常量。这个常量是 ABAP 字典中类型的域为 WDR_POPUP_BUTTON_KIND 的常量，该域中有一组固定值。固定值代表所有的有意义的可能性对话框按钮组合，如 OK、OK/Cancel 或 Yes/No/Cancel 的常数，作为 IF_WD_WINDOW 接口的属性，

默认值是 5。这意味着，该组合包含常量 CO_BUTTON_YES、CO_BUTTON_NO 和 CO_BUTTON_CANCEL，而且在对话框中将创建 3 个按钮，分别代表“Yes”“No”和“Cancel”。

（2）窗体对话框

在对话框视图的 WDDOINIT 方法中，要将按键常数链接到适当的动作 Action。为了这个目的，可使用接口 IF_WD_WINDOW 中的 SUBSCRIBE_TO_BUTTON_EVENT 方法。相应动作都必须在该对话框中直接创建，自动创建的事件处理程序方法必须进行相应的编程。

```
method wddoinit .
  data:
  l_api type ref to if_wd_view_controller,
  l_window_ctlr type ref to if_wd_window_controller,
  l_popup type ref to if_wd_window.
  l_api = wd_this->wd_get_api( ).
  l_window_ctlr = l_api->get_embedding_window_ctlr( ).
  if l_window_ctlr is bound.
    l_popup = l_window_ctlr->get_window( ).
    if l_popup is bound.
      l_popup->subscribe_to_button_event(
      button = if_wd_window=>co_button_yes
      button_text = 'Yes'                                   "#EC *
      action_name = 'YES'
      action_view = l_api
      is_default_button = abap_true ).
      l_popup->subscribe_to_button_event(
      button = if_wd_window=>co_button_no
      button_text = 'No'                                    "#EC *
      action_name = 'NO'
      action_view = l_api
      is_default_button = abap_true ).
      l_popup->subscribe_to_button_event(
      button = if_wd_window=>co_button_cancel
      button_text = 'Cancel'                                "#EC *
      action_name = 'CANCEL'
      action_view = l_api
      is_default_button = abap_true ).
    endif.
  endif.
endmethod.                          "wddoinit

      action_name = 'CANCEL'
      action_view = l_api
      is_default_button = abap_true ).
    endif.
  endif.
endmethod.                          "wddoinit
```

注：该对话框阶段模型的实例被连接到相同组件的调用窗体的实例中。出于这个原因，打开对话框时，不仅所有对话框中视图的钩子方法被调用，嵌入窗体的所有视图的钩子方法也被调用。

（3）WDDOONOPEN 和 WDDOONCLOSE 方法

每个窗体控制器具有 WDDOONOPEN 和 WDDOONCLOSE 钩子方法。这两个方法都只能在一个窗体作为一个对话框被打开或关闭时处理。由于打开一个对话框不涉及任何导航，没有入站插头被调用，因此没有事件处理程序方法被处理。WDDOONOPEN 方法可以用于实施初始化。

7.3.2 调用使用组件的对话框

如果对话框要显示的是一个通用画面，则该对话框可以用在许多不同的组件，程序员可以创建一个单独的组件，然后在相应的窗体中使用该对话框。在这种情况下，在运行时创建对话框，在当前的组件中使用方法 CREATE_WINDOW_FOR_CMP_USAGE 并将接口视图和组件使用的名称 z 作为传递参数。前提是该对话框所属的组件的属性表中必须有一个组件使用的条目。

```
method onactionpopup2_1 .
  data: l_cmp_api type ref to if_wd_component,
  l_window_manager type ref to if_wd_window_manager.
  l_cmp_api = wd_comp_controller->wd_get_api( ).
  l_window_manager = l_cmp_api->get_window_manager( ).
  if wd_this->m_popup2_1 is initial.
    wd_this->m_popup2_1 = l_window_manager->create_window_for_cmp_usage(
    interface_view_name = 'MAIN'
    component_usage_name = 'USAGE_POPUP2_1' ).
  endif.
  wd_this->m_popup2_1->open( ).
endmethod.                              "ONACTIONPOPUP2_1
```

与单独组件的对话框不同，在调用视图控制器的方法中，不能创建一个打开的对话框的按钮。因为按钮只能由所属的组件设置。这一步是在声明对话框使用的其他组件的窗体控制器的钩子函数 WDDOONOPEN 中实现的。

下面的源代码的方法 WDDOOPEN，实现控制器上的对话框：

```
method wddoonopen .
  if window_descr->is_popup = abap_true.
    window_descr->window->set_button_kind(
    if_wd_window=>co_buttons_yesnocancel ).
    window_descr->window->set_message_type(
    if_wd_window=>co_msg_type_question
    ).
  endif.
endmethod.                              "wddoonopen
```

注：如果该对话框不是独立的组件的一部分，但在组件使用中被创建，则两个窗体的阶段模型实例无法连接到相同的组件。如果该对话框是开放的，则只有中嵌入对话框中视图的钩子方法被处理。

7.3.3 调用确认对话框

要快速创建一个确认对话框的标准布局（如确认当前数据的变化），程序员可以调用接口 IF_WD_WINDOW_MANAGER 的 CREATE_POPUP_TO_CONFIRM 的方法。该方法不需要创建一个单独的窗体，运行时会自动创建该对话框。

使用 CREATE_POPUP_TO_CONFIRM 方法创建对象的类型为 IF_WD_WINDOW，对话框可以使用它的参数。

```
method onactionpopup4_1 .

  data: l_cmp_api           type ref to if_wd_component,
        l_window_manager    type ref to if_wd_window_manager,
        l_popup             type ref to if_wd_window,
        l_text              type string_table,
        l_api               type ref to if_wd_view_controller.


  l_cmp_api        = wd_comp_controller->wd_get_api( ).
  l_window_manager = l_cmp_api->get_window_manager( ).
  insert `Data where changed` into table l_text.     "#EC *
  insert `Do you want to save?`       into table l_text.     "#EC *

  l_popup = l_window_manager->create_popup_to_confirm(
              text            = l_text
              button_kind     = if_wd_window=>co_buttons_yesnocancel
              message_type    = if_wd_window=>co_msg_type_question
              window_title    = 'Test: Popup to confirm'
              window_position = if_wd_window=>co_center )."#EC *

  l_api = wd_this->wd_get_api( ).

  l_popup->subscribe_to_button_event(
             button            = if_wd_window=>co_button_yes
             action_name       = 'YES'
             action_view       = l_api
             is_default_button = abap_true ).
  l_popup->subscribe_to_button_event(
             button            = if_wd_window=>co_button_no
             action_name       = 'NO'
             action_view       = l_api
             is_default_button = abap_false ).
```

```
        l_popup->subscribe_to_button_event(
                    button              = if_wd_window=>co_button_cancel
                    action_name         = 'CANCEL'
                    action_view         = l_api
                    is_default_button = abap_false ).

        l_popup->open( ).
endmethod.
```

接口 IF_WD_WINDOW 的方法 SUBSCRIBE_TO_BUTTON_EVENT 分配相应的按钮到动作。如果已经为对话框创建了一个窗体，在钩子方法 WDDOINIT 中将动作分配给按钮。

- 如果没有明确创建窗体的存在，则相应的操作是建立在同一个视图中调用对话框。分配使用的方法 SUBSCRIBE_TO_BUTTON_EVENT 在创建的对话框后调用。
- 对于一些特殊情况，它可以用来对组件控制器或定制控制器的对话框上的按钮事件注册事件处理程序。在这些情况下，接口 IF_WD_POPUP_ TO_CONFIRM_N 的特殊属性 POPUP_TO_CONFIRM 在 IF_WD_WINDOW 接口中被实施。这个接口提供了精确的方法，组件和定制的控制器中的事件处理程序可以注册到对话框事件中。

1）选择组件 ZWD_STUDENT 中的视图 V_STUDENT_DEL，选择“Attributes”选项卡，添加属性 FLAG_DEL，如图 7-40 所示。

View V_STUDENT_DEL Active(revised)

Properties | Layout | Inbound Plugs | Outbound Plugs | Context | Attributes | Actions | Methods

Attribute	RefTo	参考打印	Description
WD_CONTEXT	☑	IF_WD_CONTEXT_NODE	Reference to Local Controller Co
WD_THIS	☑	IF_V_STUDENT_DEL	Self-Reference to Local Controll
WD_COMP_CONTROLLER	☑	IG_COMPONENTCONTROLLER	
NEXT_OPT	☐	STRING	操作
FLAG_DEL	☐	WDY_BOOLEAN	删除确认
	☐		

图 7-40

选择“Actions”选项卡，添加动作 YES、NO、CANCEL，如图 7-41 所示。

View V_STUDENT_DEL Active(revised)

Properties | Layout | Inbound Plugs | Outbound Plugs | Context | Attributes | Actions | Methods

Action	Action Type	Description	Event Handler
CONFIRM	Standard	确定更改对象	ONACTIONCONFIRM
SAVE	Standard	学生信息保存	ONACTIONSAVE
YES	Standard	删除	ONACTIONYES
NO	Standard	不删除	ONACTIONNO
CANCEL	Standard	取消	ONACTIONCANCEL
	Standard		

图 7-41

动作 YES 对应处理程序编辑如下：

```
METHOD ONACTIONYES .

  WD_THIS->FLAG_DEL = ABAP_TRUE.
ENDMETHOD.
```

动作 NO 对应处理程序编辑如下：

```
METHOD ONACTIONNO .

  WD_THIS->FLAG_DEL = ABAP_FALSE.
ENDMETHOD.
```

动作 CANCEL 对应处理程序编辑如下：

```
METHOD ONACTIONCANCEL .

  WD_THIS->FLAG_DEL = ABAP_FALSE.
ENDMETHOD.
```

选择“Methods”选项卡，添加方法 POPUP_TO_CONFIRM，如图 7-42 所示。

View V_STUDENT_DEL Active(revised)

Properties | Layout | Inbound Plugs | Outbound Plugs | Context | Attributes | Actions | Methods

Method	Method Type	Description	Event	Controller
HANDLEFROM_WINDOW	Event H...	从窗体到视图		
ONACTIONCONFIRM	Event H...	确定更改对象		
ONACTIONSAVE	Event H...	学生信息保存		
WDDOAFTERACTION	Method			
WDDOBEFOREACTION	Method			
WDDOEXIT	Method	Cleanup Method of Controller		
WDDOINIT	Method	Initialization Method of Controller		
WDDOMODIFYVIEW	Method	Method for Modifying the View Before Rendering		
WDDOONCONTEXTMENU	Method			
POPUP_TO_CONFIRM	Method	删除确认画面		
	Method			

图 7-42

方法 POPUP_TO_CONFIRM 编辑如下：

```
METHOD POPUP_TO_CONFIRM .
  DATA: L_CMP_API TYPE REF TO IF_WD_COMPONENT,
L_WINDOW_MANAGER TYPE REF TO IF_WD_WINDOW_MANAGER,
L_POPUP TYPE REF TO IF_WD_WINDOW,
L_TEXT TYPE STRING_TABLE,
L_API TYPE REF TO IF_WD_VIEW_CONTROLLER.

*   取得窗体控制器
```

```
  L_CMP_API = WD_COMP_CONTROLLER->WD_GET_API( ).
  L_WINDOW_MANAGER = L_CMP_API->GET_WINDOW_MANAGER( ).

*  编辑警示 Text
  INSERT '确认！ ' INTO TABLE L_TEXT.                         "#EC *
  INSERT '确实要删除吗？ ' INTO TABLE L_TEXT.                 "#EC *

*  创建窗体
  L_POPUP = L_WINDOW_MANAGER->CREATE_POPUP_TO_CONFIRM(
  TEXT = L_TEXT
  BUTTON_KIND = IF_WD_WINDOW=>CO_BUTTONS_YESNOCANCEL
  MESSAGE_TYPE = IF_WD_WINDOW=>CO_MSG_TYPE_QUESTION
  WINDOW_TITLE = '删除确认'
  WINDOW_POSITION = IF_WD_WINDOW=>CO_CENTER ).                  "#EC *

*  取得视图控制器
  L_API = WD_THIS->WD_GET_API( ).

*  编辑按钮
  L_POPUP->SUBSCRIBE_TO_BUTTON_EVENT(
  BUTTON = IF_WD_WINDOW=>CO_BUTTON_YES
  ACTION_NAME = 'YES'
  ACTION_VIEW = L_API
  IS_DEFAULT_BUTTON = ABAP_TRUE ).
  L_POPUP->SUBSCRIBE_TO_BUTTON_EVENT(
  BUTTON = IF_WD_WINDOW=>CO_BUTTON_NO
  ACTION_NAME = 'NO'
  ACTION_VIEW = L_API
  IS_DEFAULT_BUTTON = ABAP_FALSE ).
  L_POPUP->SUBSCRIBE_TO_BUTTON_EVENT(
  BUTTON = IF_WD_WINDOW=>CO_BUTTON_CANCEL
  ACTION_NAME = 'CANCEL'
  ACTION_VIEW = L_API
  IS_DEFAULT_BUTTON = ABAP_FALSE ).
*  弹出窗体
  L_POPUP->OPEN( ).

ENDMETHOD.
```

修改动作 SAVE 处理程序如下（删除之前确认）：

```
METHOD ONACTIONSAVE .
  DATA LO_COMPONENTCONTROLLER TYPE REF TO IG_COMPONENTCONTROLLER .
  LO_COMPONENTCONTROLLER =   WD_THIS->GET_COMPONENTCONTROLLER_CTR( ).
  DATA LS_STUDENT TYPE ZMNUMBER_T.
  LO_COMPONENTCONTROLLER->GET_STUDENT(
```

```
      IMPORTING
        STUDENT =     LS_STUDENT                              " zmnumber_t
    ).
    DATA LO_ND_NODE_CONFIG TYPE REF TO IF_WD_CONTEXT_NODE.
    DATA LO_EL_NODE_CONFIG TYPE REF TO IF_WD_CONTEXT_ELEMENT.
    DATA LS_NODE_CONFIG TYPE WD_THIS->ELEMENT_NODE_CONFIG.

*   navigate from <CONTEXT> to <NODE_CONFIG> via lead selection
    LO_ND_NODE_CONFIG = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->
WDCTX_NODE_CONFIG ).

*   get element via lead selection
    LO_EL_NODE_CONFIG = LO_ND_NODE_CONFIG->GET_ELEMENT( ).

    CASE WD_THIS->NEXT_OPT.
      WHEN 'ADD'.
        INSERT ZMNUMBER_T FROM LS_STUDENT.
        LS_NODE_CONFIG-BODY_READONLY = ABAP_FALSE.
        LS_NODE_CONFIG-KEY_READONLY     = ABAP_FALSE.
        LS_NODE_CONFIG-BODY_VISIBILE   =   CL_WD_GROUP=>E_VISIBLE-VISIBLE.
        LS_NODE_CONFIG-CONFIRM_VISIBILE = CL_WD_GROUP=>E_VISIBLE-NONE .
        LS_NODE_CONFIG-SAVE_VISIBILE   = CL_WD_GROUP=>E_VISIBLE-VISIBLE.
      WHEN 'UPD'.
        UPDATE   ZMNUMBER_T FROM LS_STUDENT.
        LS_NODE_CONFIG-BODY_READONLY = ABAP_FALSE.
        LS_NODE_CONFIG-KEY_READONLY     = ABAP_FALSE.
        LS_NODE_CONFIG-BODY_VISIBILE   =   CL_WD_GROUP=>E_VISIBLE-NONE.
        LS_NODE_CONFIG-CONFIRM_VISIBILE = CL_WD_GROUP=>E_VISIBLE-VISIBLE.
        LS_NODE_CONFIG-SAVE_VISIBILE   = CL_WD_GROUP=>E_VISIBLE-NONE.
      WHEN OTHERS.
*         删除确认
        WD_THIS->POPUP_TO_CONFIRM(
        ).
*         确认删除
        CHECK WD_THIS->FLAG_DEL = ABAP_TRUE.
        DELETE   ZMNUMBER_T FROM LS_STUDENT.
        LS_NODE_CONFIG-BODY_READONLY = ABAP_TRUE.
        LS_NODE_CONFIG-KEY_READONLY     = ABAP_FALSE.
        LS_NODE_CONFIG-BODY_VISIBILE   =   CL_WD_GROUP=>E_VISIBLE-NONE.
        LS_NODE_CONFIG-CONFIRM_VISIBILE = CL_WD_GROUP=>E_VISIBLE-VISIBLE.
        LS_NODE_CONFIG-SAVE_VISIBILE   = CL_WD_GROUP=>E_VISIBLE-NONE.

    ENDCASE.

*   set all declared attributes
    LO_EL_NODE_CONFIG->SET_STATIC_ATTRIBUTES(
```

```
                STATIC_ATTRIBUTES = LS_NODE_CONFIG ).

*   解锁
    CALL FUNCTION 'DEQUEUE_EZMNUMBER_T'
      EXPORTING
        MODE_ZMNUMBER_T = 'E'
        MANDT               = SY-MANDT
        ZZNUMBER            = LS_STUDENT-ZZNUMBER
        X_ZZNUMBER          = ' '
        _SCOPE              = '3'
        _SYNCHRON           = ' '
        _COLLECT            = ' '.

ENDMETHOD.
```

2）选择组件应用 zwd_student_del，运行结果，如图 7-43 所示。



图 7-43

7.4　输入帮助

在交互式业务应用程序的环境中，由用户输入的数据是非常重要的。在大量的对话框变式中，一个有效的输入表示一个有限制的一组值的一个元素。用户可以从一个包含可能值的表中选择所需的值。

注：输入帮助也被称为值帮助、搜索帮助、F4 帮助等。

（1）Web Dynpro ABAP 的输入帮助

一旦输入帮助集成的 Web Dynpro 框架自动生成，就实现关联图标，以便用户在运行时选

择调用输入帮助。同时，F4 键会自动分配给系统，用于调用输入帮助。

有许多不同类型的输入帮助。用户选择哪种类型的输入帮助依赖于相应的应用程序。

Web Dynpro ABAP 框架提供了两种实现输入帮助的方法：

1）ABAP 字典搜索帮助。该输入帮助是基于 SAP 系统的 ABAP 字典中的搜索帮助功能。因此用户可以使用 ABAP 字典中的数据元素搜索帮助的定义，只要这些支持 Web Dynpro。

2）OVS 值帮助。

如果不能使用 ABAP 词典的搜索帮助，例如，如果要根据 Web Dynpro 应用程序中输入的值的计算结果进行搜索，用搜索帮助不能实现需求，则可以使用对象值选择（Object Value Selection）输入帮助，即 OVS 值帮助。

这两种预实现类型的输入帮助具有的优点是可以在同一时间输入几个字段进行搜索。

此外，程序员也可以编写自己的输入帮助。程序员也可以利用选择标准的选择选项（Select Options.），为输入帮助添加更多的限制。

程序员也可以使用一个下拉列表，即 UI 元素 DropDownByIndex 或 UI 元素 DropDown ByKey。若用户使用这些 UI 元素，则应该保持设定的值的范围相对较小。

注：太多的下拉式列表框在一个视图中，可能会导致性能瓶颈问题。

在 Web Dynpro ABAP 应用程序中, 用户可以从 ABAP 字典中重复使用现有的搜索帮助。不过，也有数据类型系统中没有提供搜索帮助。例如，TIMS 数据类型或 T（时间）没有 F4 帮助。

应用程序可以通过查询类 CL_WEB_DYNRPO 的属性的 is_active，找出它是否正在 Web Dynpro 环境中运行。

语法：

```
IF CL_WEB_DYNPRO=>IS_ACTIVE = ABAP_TRUE.

   ...

ELSE.

   ...

ENDIF.
```

在视图中可以创建输入的字段的帮助（F4 帮助），这使得它更容易为用户输入数据。InputField 元素结束时的上传提示符号意味着搜索帮助可用。

注：搜索帮助符号始终是相同的，无论是来自 ABAP 字典的搜索帮助，还是 OVS 搜索帮助，或应用程序中自定义的搜索帮助，如图 7-44 所示。

示例：

（2）特点

有以下两种类型的搜索帮助：

● 基本搜索帮助。
● 复合搜索帮助。

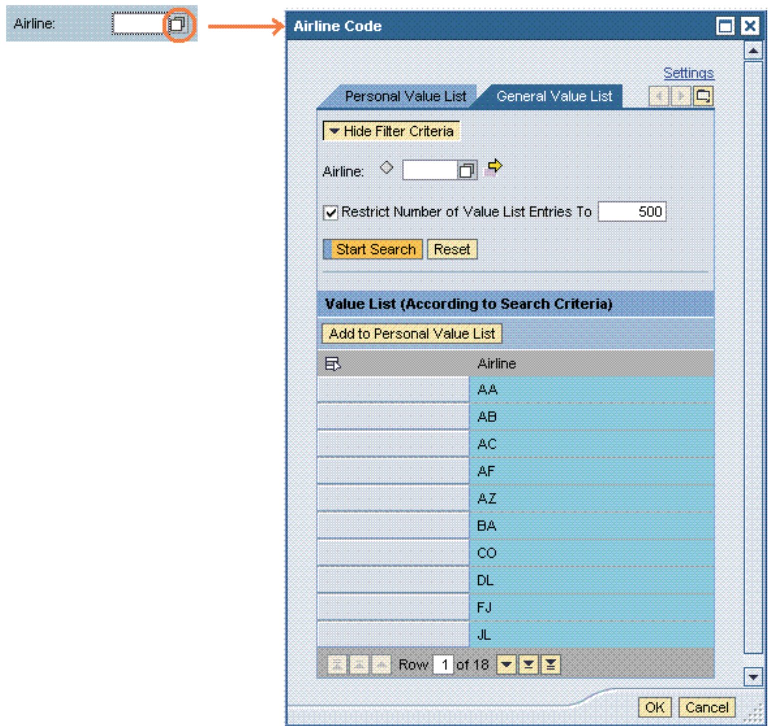


图 7-44

注：程序员编辑搜索帮助：为了确保最佳的用户指导，程序员可以通过在搜索帮助增强中修改搜索帮助弹出时间点禁止关闭弹出值帮助。一个打开的对话框中，只能由用户通过以下行动关闭。

● 从值列表中选择一个值
● 触发“取消”按钮或

用户期望相关的值列表，即使它是空的，也不希望帮助对话框被关闭。

（3）日期字段的值帮助

标准值的一周的第一天是星期一。用户可以 BADI（CALENDAR_DEFINITION）定义不同的一天，由此显示一周的天数和周数变化计算：

日期字段的值帮助（功能模块 F4_DATE 的功能组 SHLC）调用以上的 BADI。在 F4 的帮助中，一周的第一天是从 BADI 实施改变后其中一周中的一天。如果周数偏离 ISO 规则，则 F4 帮助隐藏这一点，因为所使用的前端控制是不正常的周数。

（4）集成

ABAP 字典搜索帮助通过 Context 属性的 Value Help Mode 属性集成到 Web Dynpro 应用程序中。

注：Context 节点必须参照一个字典结构定义。这确保了结构的所有部件可在运行时作为该节点的动态属性。

只有选项为 Automatic 和 Dictionary Search Help 与 ABAP 字典搜索帮助相关联。

1）Automatic。搜索帮助被分配到参照 ABAP 数据字典的 Context 属性的数据类型。

示例：

ABAP 数据字典定义，如图 7-45 所示。

数据元素	ZZNUMBER_E	激活
短文本	学生学号	

Attributes | Data Type | Further Characteristics | Field Label

Search Help	
Name	ZZMCSNAME
Parameters	ZZNUMBER

图 7-45

Context 属性定义，如图 7-46 所示。

Change Attribute

Attribute Name	ZZNUMBER
Type assignment	Type
Type	ZMNUMBER_T-ZZNUMBER
Read-only	No
Default Value	
Input Help Mode	Automatic
Ermittelte Wertehilfe	ZZMCSNAME
Type of Input Help	搜索帮助

图 7-46

如果没有搜索帮助用于 Context 的属性类型，就没有搜索帮助符号显示，也没有可以执行的搜索帮助。

2）Dictionary Search Help。ABAP 字典中的一个搜索帮助可以分配到 Context 属性。在属性 Input Help Mode 字段中选择 Dictionary Search Help 栏位时，程序员可以在其下面的文本框中输入需要的搜索帮助。在 ABAP 字典中，必须事先定义搜索帮助。如果不输入搜索帮助，则在运行时，系统会尝试使用 Context 属性的类型确定的搜索帮助。

示例：

Context 属性定义，如图 7-47 所示。

Create Attribute

Attribute Name	TEXTID
Type assignment	Type
Type	STRING
Read-only	No
Default Value	
Input Help Mode	Dictionary Search Help
Dictionary-Suchhilfe	ZZMCSBID

Additional Attribute

图 7-47

注：搜索帮助的输入和输出参数必须在节点同一范围内确定。

（5）约束

- 复合搜索帮助不支持热键。
- 搜索帮助增强含有 DYNPRO 特定的功能，如 CALL SCREEN 不支持。
- 元素 TIMS 没有自动的 F4 帮助。
- Web Dynpro ABAP 的日期值帮助不支持所有 ABAP 日期格式。

7.4.1 OVS 输入帮助

F4 输入帮助的方法之一是 OVS 值帮助（对象值选择 Object Value Selection）。OVS 值帮助是在系统端的 Web Dynpro 组件中实现的。

当 OVS 值帮助已定义一个 Context 属性后，它在绑定到这个属性的所有输入框时是可用的。在运行时，当用户按下〈F4〉键或单击值帮助图标时，OVS 组件自动实例化。同时，该对话框会自动创建在屏幕上。

1. 系统组件 WDR_OVS

在视图上组件 WDR_OVS 提供了搜索结果可以被显示为一个表的功能。该组件还包含一个选择视图，可以用来限制搜索条件。

视图中的输入字段以及要显示的表的结构和内容都要在应用程序中定义。OVS 组件因此在一个合适的时间需要使用组件的回调。

2. 事件 OVS 及参数 OVS_CALLBACK_OBJECT

在 OVS 组件的接口控制器中使用组件的回调函数通过 OVS 事件实现。该事件一前一后被自动触发 4 次。它传递类型为 IF_WD_OVS 的参数 OVS_CALLBACK_OBJECT 到相应应用程序组件的事件处理程序中。事件参数实例属性 PHASE_INDICATOR 确定触发阶段。第一次触发该事件时的属性值为 CO_PHASE_0，当它被触发第二次时值为 CO_PHASE_1，以此类推。下面详细描述 4 个时间点，如图 7-48 所示。

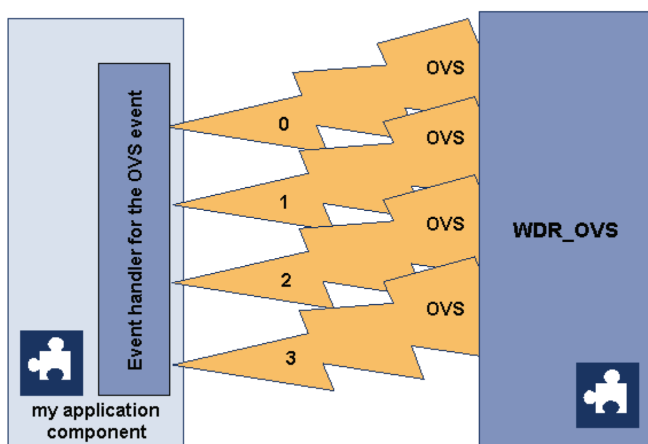


图 7-48

（1）PHASE_INDICATOR = IF_WD_OVS => CO_PHASE_0

此时，可以配置 OVS 组件。例如，窗体标题、搜索帮助列表标题，可以定义输出表的列标题。此外，还可以设置在事件表中是否应选择一行或多个行。为实现这一目的，事件参数 OVS_CALLBACK_OBJECT 提供了方法 SET_CONFIGURATION，它可以在单独的时间点上使用，也可以在以后的某个时间点上调用此方法触发一个错误消息。

该阶段又称为配置阶段。此处，用户可以定义 OVS 组件在选择屏幕对话框或输入帮助列表中显示的文本，还可以定义组成选择对话框的列的数量。要将信息导出至 OVS 组件，需调用回调对象的方法 set_configuration()。

注：如果 OVS 组件不显示选择对话框，则可以忽略阶段 0 的调用方法 set_configuration( )。

如果据此输入相关输入字段，则无法从 ABAP 字典获取用户组件不提供的标签文本。

（2）PHASE_INDICATOR = IF_WD_OVS => CO_PHASE_1

用户可以定义选择对话框中要显示的输入字段。可通过导出包含这些字段的任意结构来完成该操作。通过将结构字段设置为相应值，可指定选择字段的默认值。

要将信息导出至 OVS 组件，需调用回调对象的方法 set_input_structure()，然后可呈现选择屏幕。

注：可以忽略阶段 1 的调用方法 set_input_structure( )。这种情况下，将不显示选择屏幕（直接显示列表对象）。如果用户已经在用户组件的输入字段中输入了值，则在显示输入帮助之前，不需要再次在 OVS 选择对话框上显示相同的值。

（3）PHASE_INDICATOR = IF_WD_OVS => CO_PHASE_2

在此阶段，在用户按下 OVS 选择对话框（如果已处理）的按钮后，将开始处理该阶段。在此阶段，用户必须收集 OVS 组件稍后将显示为输入帮助列表的数据。然而，要收集这些信息，用户组件需要了解用户已经在选择对话框的字段中输入的内容。回调对象的 query_parameters

参数可提供此信息的引用（数据引用）。

要将值列表导出至 OVS 组件，需调用回调对象的方法 set_output_table( )，随后显示该值列表。

（4）PHASE_INDICATOR = IF_WD_OVS => CO_PHASE_3

在选择值列表的值后，必须将该选择传输回用户组件。因此，需要第四次触发 OVS 事件。数据引用变量 selection 是回调对象 OVS_CALLBACK_OBJECT 的参数，可提供用户选择的引用。在用户组件事件处理器方法 SET_CONFIGURATION 的相关编码段中，必须将这些值写回至 Context。选定数据显示为用户组件视图的输入字段值。

注：OVS 具有 4 个阶段，且每个阶段只调用一次。

可使用下列属性识别不同的阶段：OVS_CALLBACK_OBJECT-> PHASE_INDICATOR。

3. 绑定值到一个 Context 属性

要绑定一个 OVS 帮助的值到 Context 属性，需执行以下步骤：

1）组件 WDR_OVS 的组件用法必须由用户组件声明。

2）OVS 界面控制器的用法必须在用户组件的控制器中声明，如组件控制器或显示视图控制器。

3）必须为相关的属性选择输入帮助类型为对象值选择器（Object Value Selection）。

4）必须创建并实施 OVS 组件用法的事件 OVS 处理器方法。

4. OVS 值帮助 Context 更改日志

当用户使用 OVS 值帮助时，仍然可以使用 OVS 值帮助 Context 更改日志的功能。

注：在阶段 PHASE_INDICATOR IF_WD_OVS => CO_PHASE_3 中，所选择的数据必须显式输入更改日志的表中，所以这仅在需要的地方会自动为用户输入。

7.4.2 自定义输入帮助

除了上述输入帮助技术外，程序开发员还可定义一个完全由用户定义的输入帮助。从技术上讲，该类输入帮助可作为实施 Web Dynpro 组件界面 IWD_VALUE_HELP 的 Web Dynpro 组件来实施。一旦用户自定义的输入帮助链接到 Context 属性，则视图中任何绑定 Context 属性的输入框的输入帮助会变得可用。

（1）链接到 Context 属性

要使用某个输入字段的输入帮助，需要执行下列步骤：

1）输入帮助组件的组件用法必须由用户组件声明。

2）输入帮助组件的界面控制器的用法必须在用户组件视图中声明。

3）必须为相关的属性选择输入帮助模式用户定义的编程。

（2）接口 IWD_VALUE_HELP

输入帮助组件的组件接口有一个方法：set_value_help_listener( )。如果单击相关输入字段的输入帮助按钮，则 Web Dynpro 运行时环境会调用该方法。

必须按下列步骤实施输入帮助组件：

- 方法 set_value_help_listener( ) 有一个导入参数。这意味着 WebDynpro 运行时，环境

所提供的收听器引用将传输到用户定义的输入帮助组件。帮助组件可以使用该接口告诉框架窗体被关闭。例如，如果数据已被选择，或者取消已被按下，或者发生了一个错误。该接口包含两个属性 F4_CONTEXT_ELEMENT 和 F4_ATTRIBUTE_INFO。它们描述的 F4 帮助已请求 Context 元素。必须将该引用保存为用户定义的控制器属性。

- 事件 VH_WINDOW_CLOSED 和 VH_WINDOW_OPENED 会触发 Web Dynpro 框架。这些事件表明的时间点为输入帮助窗体的关闭或打开。在组件接口 IWD_VALUE_HELP 实施时，这些事件被复制到值帮助组件中，这使它们可以调用组件，并允许其注册这些事件的事件处理程序到组件接口。要关闭帮助值对话框，必须使用收听器的 close_window( ) 方法。
- 必须将所有视图嵌入名为 WD_VALUE_HELP 的窗体中。Web Dynpro 运行时环境会使用该名称。
- Context 映射可用于在输入帮助组件和用户组件之间交换数据。

注：*要实施组件接口 IWD_VALUE_HELP，需要选择输入帮助组件中的重新实施按钮。实施状态变为绿色，并且输入帮助组件 HC 控制器中会显示组件界面的事件和方法。*

（3）自定义输入帮助 Context 更改日志

当用户使用自定义输入帮助时，仍然可以使用 Context 更改日志的功能。但是，由用户选择的值不会自动写入更改表，需要编程明确。

7.4.3 选择标准

组件 WDR_SELECT_OPTIONS 作为一个选项屏幕的可重用的组件，尤其是当使用输入范围作为输入帮助时。接口 IF_WD_SELECT_OPTIONS 提供用于此目的处理程序。

注：*参照经典 SAP GUI 的选择屏幕的选择选项，Web Dynpro ABAP 通常会提供相同的功能。需要注意的是，Web Dynpro ABAP 不支持逻辑数据库。*

要使用选择选项部分嵌入 WDR_SELECT_OPTIONS 组件到自己的组件。

注：*要创建一个组件的使用，而不是嵌入在选择选项组件本身的窗口中。*

4 个标准按钮提供以下功能。

1）取消设置（Cancel）：终止处理。系统返回到应用程序中的最后一个显示的画面。

2）复位（Reset）：由用户输入的所有选择字段的内容被清除。接口提供方法 RESET_SELECTION_FIELD（复位某一个选择字段的范围表）和 RESET_ALL_SELCTIONS_FIELDS（复位所有选择字段的范围表）用于此。

3）完成（Done）：确定用户输入并使用输入的数据继续处理。

4）调查（Check）：由用户输入的数据进行验证检查。接口方法 CHECK_ALL_SELECTION_FIELDS（检查，所有的选择信息的输入是正确的）设置在接口上。

对于取消和复制两个功能，用户只有在 WDR_SELECT_OPTIONS 组件中注册事件，才能实现取消和复制功能。

当用户单击其中一个按钮时，该事件被触发。注意，只有在自由定义输入帮助和 ABAP

字典输入帮助的选择选项（Select Options）是有效的，但不支持 OVS 帮助。

使用 SET_GLOBAL_OPTIONS 方法可以隐藏这些按钮中的一个或全部，并可以指定哪些标准按钮不想设置。由于单独的功能，用户还可以定义自己的按钮。

一旦需要停用标准按钮，则程序员可以调用组件接口或辅助类的相应方法，在源代码中触发的动作。

方法 ADD_SELECTION_FIELD 可以为字段增加查询条件。

注：Web Dynpro ABAP 的选择选项组件还支持 dynpro 环境中的参数字段。

组件程序 WDR_TEST_SELECT_OPTIONS 的运行结果如图 7-49 所示。

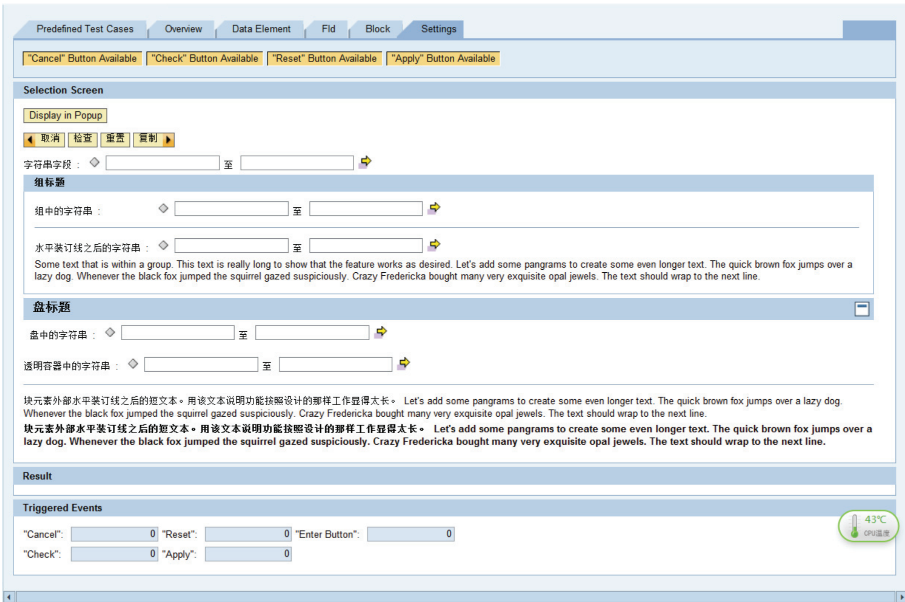


图 7-49

7.4.4 编辑 Web Dynpro 应用程序

1. 自定义检索帮助应用程序

1）在 ABAP 工作台中的 Web Dynpro 组件编辑界面创建 Web Dynpro 组件。

选中“Web Dynpro Comp./Intf.”，在相应的文本框中输入程序的名称，如图 7-50 所示。

创建 Web Dynpro 组件，如图 7-51 所示。

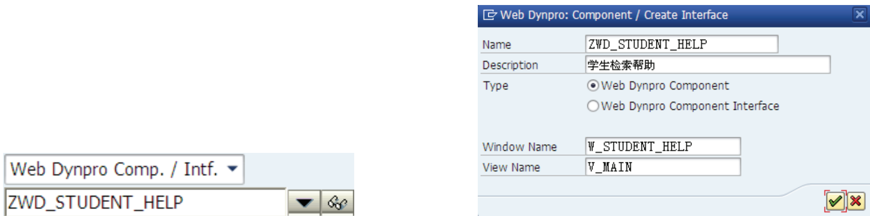


图 7-50　　图 7-51

编辑 Web Dynpro 组件，如图 7-52 所示。

图 7-52

2）在 ABAP 工作台中的组件编辑界面添加组件接口。

删除窗体 W_STUDENT_HELP，实施组件接口 IWD_VALUE_HELP，如图 7-53 所示。

图 7-53

3）在 ABAP 工作台中的组件编辑界面编辑组件控制器。

选择组件控制器 COMPONENTCONTROLLER，选择“Context”选项卡，创建 Context，节点 NODE_LIST，如图 7-54 所示。

图 7-54

节点 NODE_SELECTED，如图 7-55 所示。

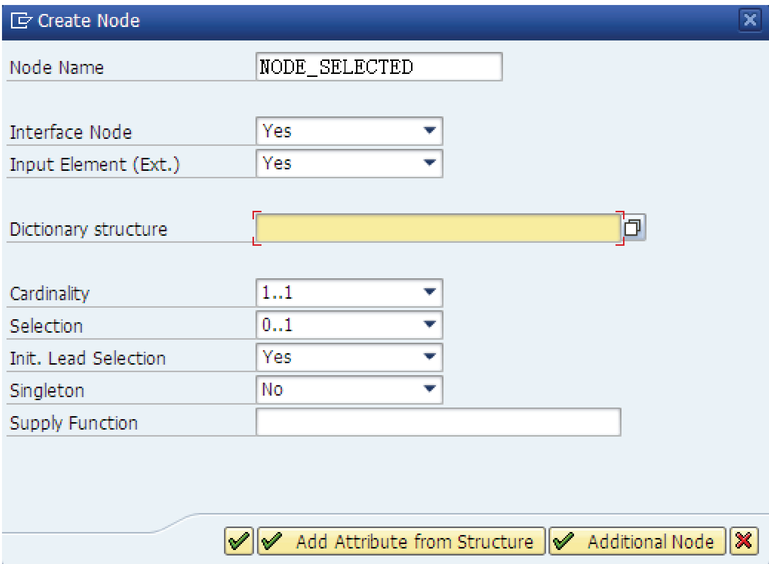


图 7-55

编辑结果如图 7-56 所示。

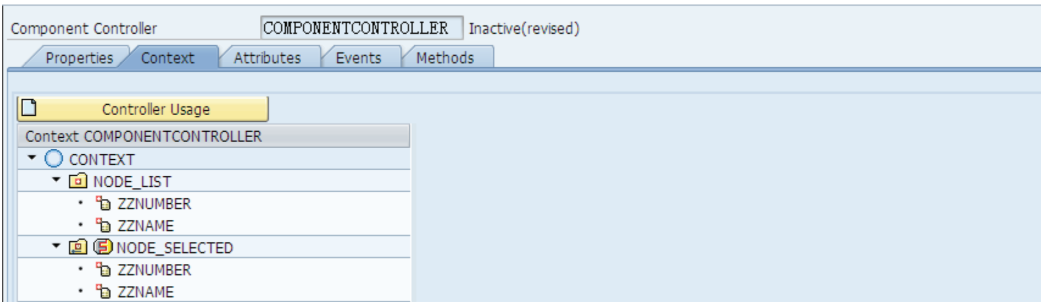


图 7-56

选择“Attributes”选项卡，添加属性如图 7-57 所示。

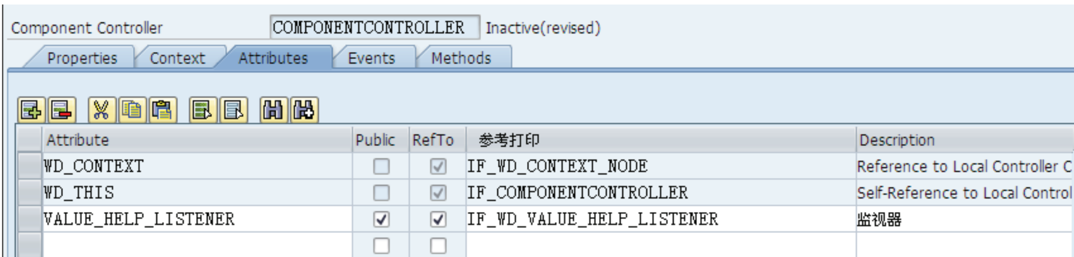


图 7-57

选择“Events”选项卡，添加事件如图 7-58 所示。



图 7-58

选择“Methods”选项卡，编辑具体方法如图 7-59 所示。

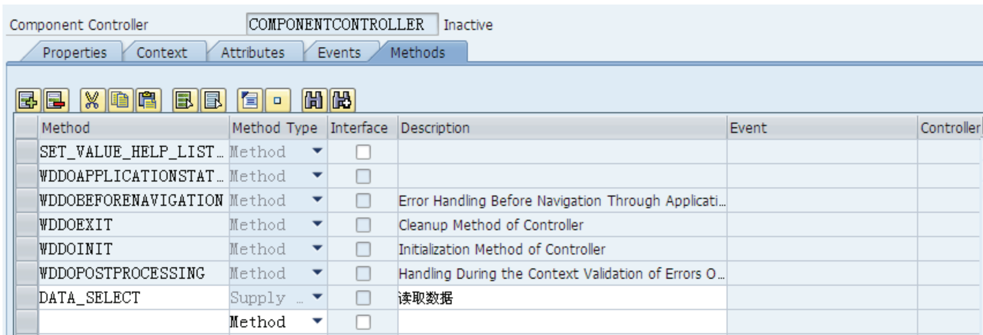


图 7-59

方法 SET_VALUE_HELP_LISTENER 编辑如下：

```
METHOD SET_VALUE_HELP_LISTENER .

   WD_THIS->VALUE_HELP_LISTENER = LISTENER.

ENDMETHOD.
```

供给函数编辑如下：

```
METHOD DATA_SELECT .
* General Notes
* =============
* A common scenario for a supply method is to aquire key
* informations from the parameter <parent_element> and then
* to invoke a data provider.
* A free navigation thru the context, especially to nodes on
* the same or deeper hierachical level is strongly discouraged,
* because such a strategy may easily lead to unresolvable
* situations!!

*   if necessary, get static attributes of parent element
*   DATA ls_parent_attributes TYPE wd_this->element_context.
```

```
*    parent_element->get_static_attributes(
*        IMPORTING
*            static_attributes = ls_parent_attributes ).

*
* data declaration
   DATA LT_NODE_LIST TYPE WD_THIS->ELEMENTS_NODE_LIST.
   DATA LS_NODE_LIST LIKE LINE OF LT_NODE_LIST.
* @TODO compute values
   SELECT    ZZNUMBER
                 ZZNAME
                 FROM    ZMNUMBER_T INTO TABLE    LT_NODE_LIST.

* bind all the elements
   NODE->BIND_TABLE(
      NEW_ITEMS                    =    LT_NODE_LIST
      SET_INITIAL_ELEMENTS = ABAP_TRUE ).

ENDMETHOD.
```

4）在 ABAP 工作台中的视图编辑界面编辑视图。

选择视图 V_MAIN，选择“Context”选项卡，映射并创建 Context 如图 7-60 所示。

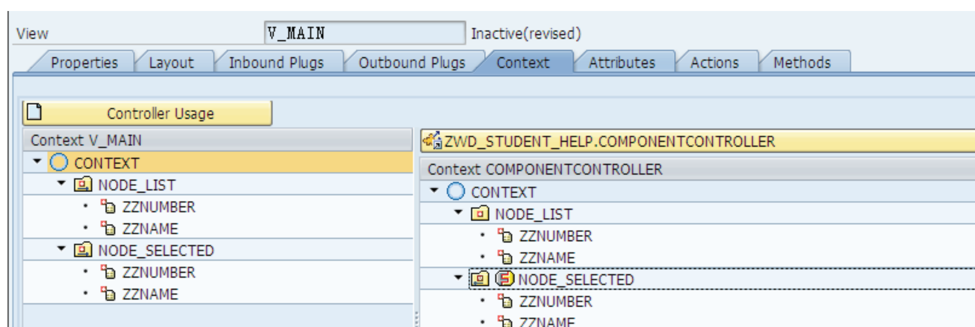


图 7-60

选择“Layout”选项卡，创建 UI 元素，如图 7-61 所示。

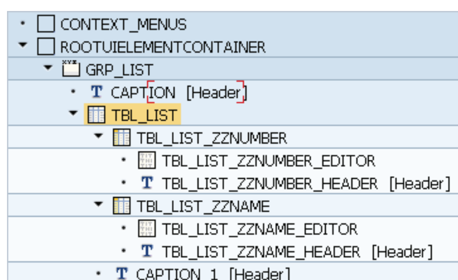


图 7-61

表 UI 元素 TBL_LIST 的 onLeadSelect 属性添加动作 DATA_SELECT，如图 7-62 所示。

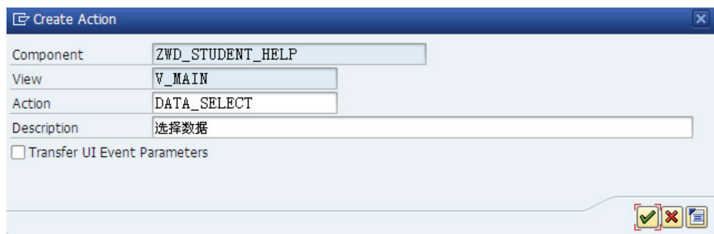


图 7-62

编辑结果如图 7-63 所示。



图 7-63

选择“Actions”选项卡，编辑动作 DATA_SELECT 的处理程序如下：

```
METHOD ONACTIONDATA_SELECT .
*   取得选择数据
  DATA LO_ND_NODE_LIST TYPE REF TO IF_WD_CONTEXT_NODE.
  DATA LO_EL_NODE_LIST TYPE REF TO IF_WD_CONTEXT_ELEMENT.
  DATA LS_NODE_LIST TYPE WD_THIS->ELEMENT_NODE_LIST.
*     navigate from <CONTEXT> to <NODE_LIST> via lead selection
  LO_ND_NODE_LIST = WD_CONTEXT->GET_CHILD_NODE( NAME =
WD_THIS->WDCTX_NODE_LIST ).
*     get element via lead selection
  LO_EL_NODE_LIST = LO_ND_NODE_LIST->GET_ELEMENT( ).
*     get all declared attributes
  LO_EL_NODE_LIST->GET_STATIC_ATTRIBUTES(
    IMPORTING
      STATIC_ATTRIBUTES = LS_NODE_LIST ).
*向外界传值
  DATA LO_ND_NODE_SELECTED TYPE REF TO IF_WD_CONTEXT_NODE.
  DATA LO_EL_NODE_SELECTED TYPE REF TO IF_WD_CONTEXT_ELEMENT.
  DATA LS_NODE_SELECTED TYPE WD_THIS->ELEMENT_NODE_SELECTED.
* navigate from <CONTEXT> to <NODE_SELECTED> via lead selection
  LO_ND_NODE_SELECTED = WD_CONTEXT->GET_CHILD_NODE( NAME =
WD_THIS->WDCTX_NODE_SELECTED ).
* get element via lead selection
```

```
    LO_EL_NODE_SELECTED = LO_ND_NODE_SELECTED->GET_ELEMENT( ).
* @TODO fill static attributes
    MOVE-CORRESPONDING LS_NODE_LIST   TO LS_NODE_SELECTED .
* set all declared attributes
    LO_EL_NODE_SELECTED->SET_STATIC_ATTRIBUTES(
       STATIC_ATTRIBUTES = LS_NODE_SELECTED ).
*   触发事件
    WD_COMP_CONTROLLER->FIRE_VH_DATA_SELECTED_EVT(
     ).
    WD_COMP_CONTROLLER->VALUE_HELP_LISTENER->CLOSE_WINDOW( ).
ENDMETHOD.
```

5）在 ABAP 工作台中的窗体编辑界面编辑窗体。

选择窗体 WD_VALUE_HELP，将视图 V_MAIN 嵌入窗体编辑如图 7-64 所示。

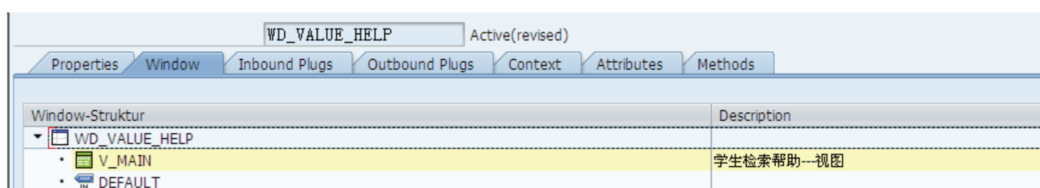


图 7-64

2. 检索帮助应用主程序

1）启动 ABAP 工作台，在 Web Dynpro 组件编辑界面创建 Web Dynpro 组件。

选中“Web Dynpro Comp./Intf.”，在相应的文本框中输入程序的名称，如图 7-65 所示。

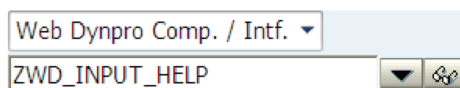


图 7-65

创建 Web Dynpro 组件如图 7-66 所示。



图 7-66

编辑 Web Dynpro 组件如图 7-67 所示。

Object Name	Description
ZWD_INPUT_HELP	学生检索
COMPONENTCONTROLLER	学生检索---组件控制器
Component Interface	
INTERFACECONTROLLEF	学生检索---组件控制器
Interface Views	
W_HELP	学生检索---窗体
Views	
V_MAIN	学生检索---视图
Windows	
W_HELP	学生检索---窗体

图 7-67

2）在 ABAP 工作台中的 Web Dynpro 组件编辑界面添加使用组件。

选择组件，选择“Used Components”选项卡，添加使用组件，如图 7-68 所示。

Web Dynpro Component: ZWD_INPUT_HELP Active/revised
Description: 学生检索
Assistance Class:
Created By: BCUSER Created On: 2011/10/24
Last Changed By: BCUSER Changed On: 2011/10/24
Original Lang.: ZH Package: ZWD_TRAINING
Accessibility Checks Active
Used Components | Implemented interfaces

Used Web Dynpro Components

Component Use	Component	Description of Component
STUDENT_OVS	WDR_OVS	
STUDENT_HELP	ZWD_STUDENT_HELP	学生检索帮助

图 7-68

3）在 ABAP 工作台中的组件控制器编辑界面编辑组件控制器。

选择组件控制器 COMPONENTCONTROLLER，选择“Properties”选项卡，添加使用组件，如图 7-69 所示。

Component Controller: COMPONENTCONTROLLER Inactive(revised)
Properties | Context | Attributes | Events | Methods

Description: 学生检索---组件控制器
创建者: BCUSER 创建日期: 2011/10/24
最后修改人: BCUSER 更改日期: 2011/10/24

Used Controllers/Components

Component Use	Component	Controller	Description
STUDENT_HELP	ZWD_STUDENT_HELP		学生检索帮助
STUDENT_HELP	ZWD_STUDENT_HELP	INTERFACECONTROLLER	学生检索帮助---组件控制器

图 7-69

选择“Context”选项卡，创建 Context，如图 7-70 所示。

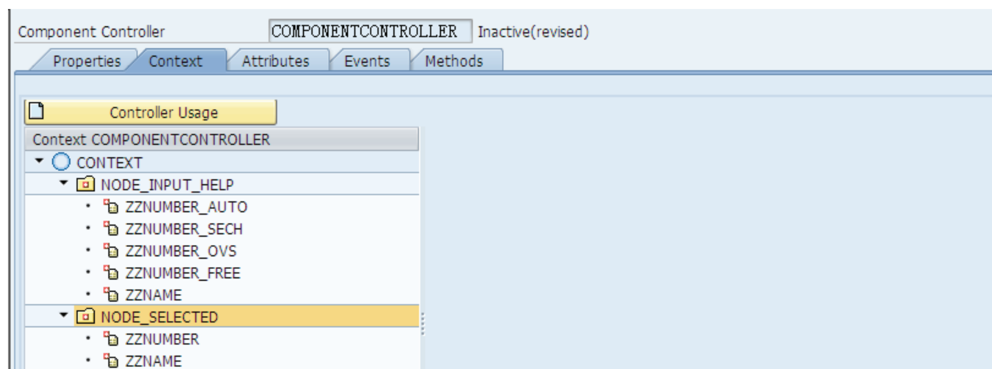


图 7-70

NODE_SELECTED 的节点 NODE_INPUT_HELP，如图 7-71 所示。

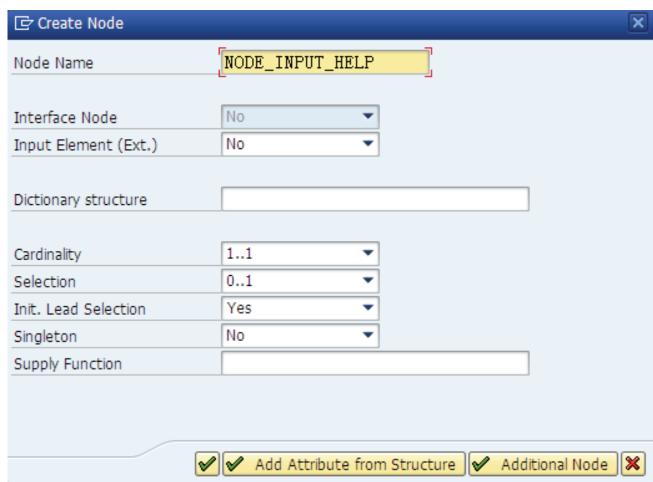


图 7-71

属性 ZZNUMBER_AUTO，如图 7-72 所示。



图 7-72

属性 ZZNUMBER_SECH，如图 7–73 所示。

Create Attribute

Attribute Name	ZZNUMBER_SECH
Type assignment	Type
Type	STRING
Read-only	No
Default Value	
Input Help Mode	Dictionary Search Help
Dictionary-Suchhilfe	ZZMCSBID

Additional Attribute

图 7–73

属性 ZZNUMBER_OVS，如图 7–74 所示。

Change Attribute

Attribute Name	ZZNUMBER_OVS
Type assignment	Type
Type	STRING
Read-only	No
Default Value	
Input Help Mode	Object Value Selector
OVS-Component-Verwendung	STUDENT_OVS

图 7–74

属性 ZZNUMBER_FREE，如图 7–75 所示。

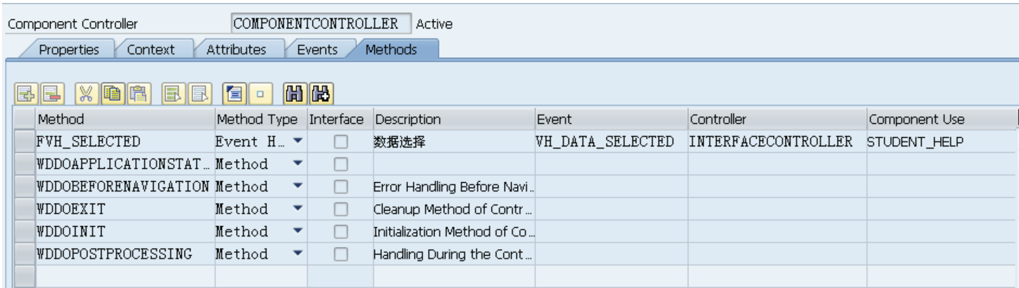
Create Attribute

Attribute Name	ZZNUMBER_FREE
Type assignment	Type
Type	STRING
Read-only	No
Default Value	
Input Help Mode	Freely Programmed
Wertehilfe-Component-Verw	STUDENT_HELP

Additional Attribute

图 7–75

选择“Methods”选项卡，编辑事件处理程序如图 7-76 所示。



Method	Method Type	Interface	Description	Event	Controller	Component Use
FVH_SELECTED	Event H…	☐	数据选择	VH_DATA_SELECTED	INTERFACECONTROLLER	STUDENT_HELP
WDDOAPPLICATIONSTAT…	Method	☐				
WDDOBEFORENAVIGATION	Method	☐	Error Handling Before Navi…			
WDDOEXIT	Method	☐	Cleanup Method of Contr…			
WDDOINIT	Method	☐	Initialization Method of Co…			
WDDOPOSTPROCESSING	Method	☐	Handling During the Cont…			

图 7-76

```
METHOD FVH_SELECTED .
  DATA LO_ND_NODE_SELECTED TYPE REF TO IF_WD_CONTEXT_NODE.

  DATA LO_EL_NODE_SELECTED TYPE REF TO IF_WD_CONTEXT_ELEMENT.
  DATA LS_NODE_SELECTED TYPE WD_THIS->ELEMENT_NODE_SELECTED.

* navigate from <CONTEXT> to <NODE_SELECTED> via lead selection
  LO_ND_NODE_SELECTED = WD_CONTEXT->GET_CHILD_NODE( NAME =
WD_THIS->WDCTX_NODE_SELECTED ).
* get element via lead selection
  LO_EL_NODE_SELECTED = LO_ND_NODE_SELECTED->GET_ELEMENT( ).
* get all declared attributes
  LO_EL_NODE_SELECTED->GET_STATIC_ATTRIBUTES(
    IMPORTING
      STATIC_ATTRIBUTES = LS_NODE_SELECTED ).
  DATA LO_ND_NODE_INPUT_HELP TYPE REF TO IF_WD_CONTEXT_NODE.

  DATA LO_EL_NODE_INPUT_HELP TYPE REF TO IF_WD_CONTEXT_ELEMENT.
  DATA LS_NODE_INPUT_HELP TYPE WD_THIS->ELEMENT_NODE_INPUT_HELP.
* navigate from <CONTEXT> to <NODE_INPUT_HELP> via lead selection
  LO_ND_NODE_INPUT_HELP = WD_CONTEXT->GET_CHILD_NODE( NAME =
WD_THIS->WDCTX_NODE_INPUT_HELP ).
* get element via lead selection
  LO_EL_NODE_INPUT_HELP = LO_ND_NODE_INPUT_HELP->GET_ELEMENT( ).
* @TODO fill static attributes
  MOVE LS_NODE_SELECTED-ZZNAME to LS_NODE_INPUT_HELP-ZZNAME.
  MOVE LS_NODE_SELECTED-ZZNUMBER to LS_NODE_INPUT_HELP-ZZNUMBER_FREE .

* set all declared attributes
  LO_EL_NODE_INPUT_HELP->SET_STATIC_ATTRIBUTES(
     STATIC_ATTRIBUTES = LS_NODE_INPUT_HELP ).

ENDMETHOD.
```

4）ABAP 工作台中的编辑界面创建组件，使用组件间的 Context 映射。

选择组件接口 INTERFACECONTROLLER_USAGE，选择“Properties”选项卡，添加使用组件，如图 7-77 所示。

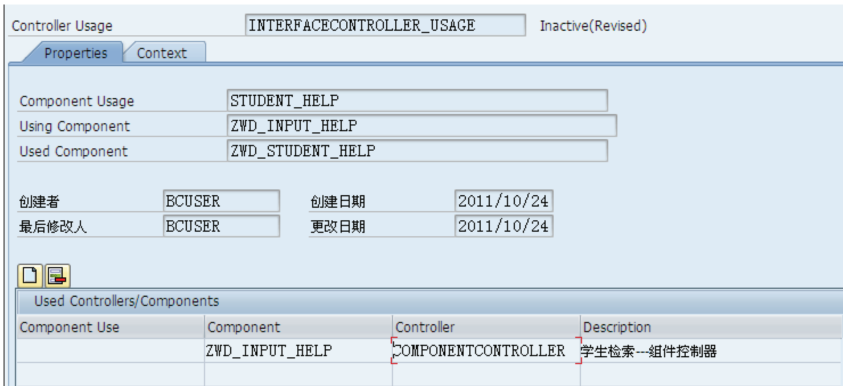


图 7-77

选择“Properties”选项卡，为组件接口 Context 建立外部映射，如图 7-78 所示。

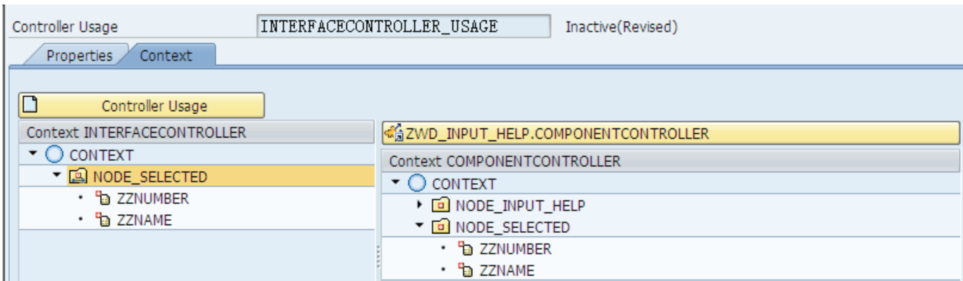


图 7-78

5）在 ABAP 工作台中的视图编辑界面编辑视图。

选择“Properties”选项卡，为视图添加组件使用如图 7-79 所示。

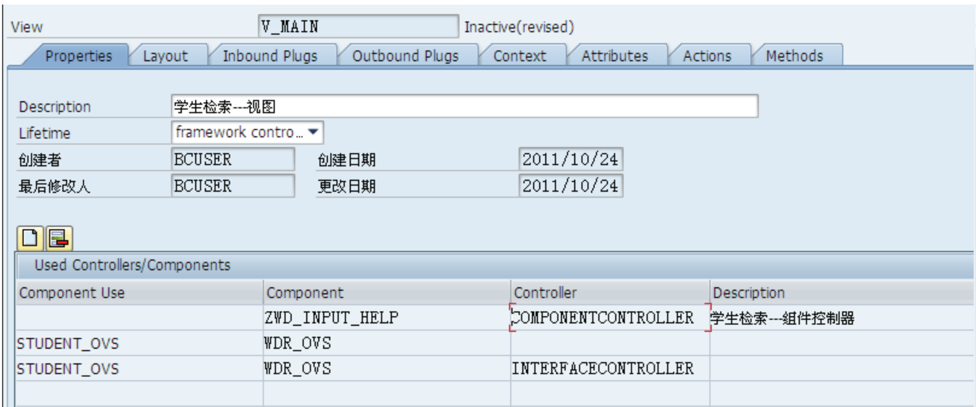


图 7-79

选择“Properties”选项卡，映射并创建 Context，如图 7-80 所示。

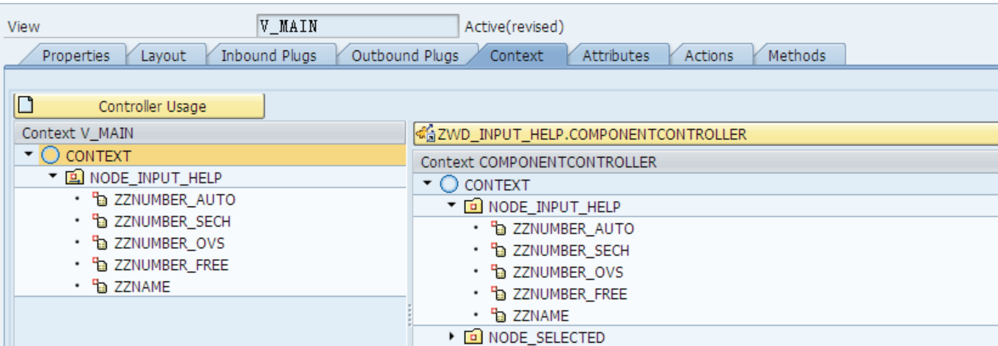
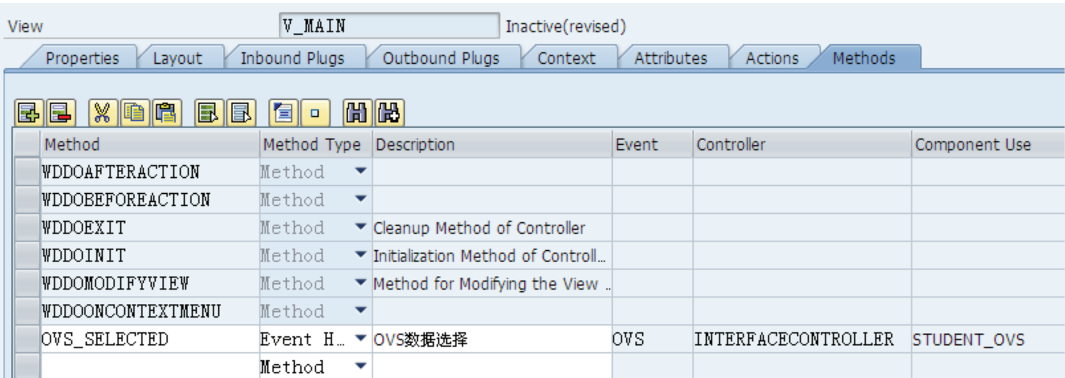


图 7-80

选择“Methods”选项卡，编辑事件处理程序如图 7-81 所示。



Method	Method Type	Description	Event	Controller	Component Use
WDDOAFTERACTION	Method				
WDDOBEFOREACTION	Method				
WDDOEXIT	Method	Cleanup Method of Controller			
WDDOINIT	Method	Initialization Method of Controll…			
WDDOMODIFYVIEW	Method	Method for Modifying the View …			
WDDOONCONTEXTMENU	Method				
OVS_SELECTED	Event H…	OVS数据选择	OVS	INTERFACECONTROLLER	STUDENT_OVS
	Method				

图 7-81

```
METHOD OVS_SELECTED .
* declare data structures for the fields to be displayed and
* for the table columns of the selection list, if necessary
  TYPES:
    BEGIN OF LTY_STRU_INPUT,
*     add fields for the display of your search input here
      ZZNUMBER TYPE ZZNUMBER_E,
      ZZNAME TYPE ZZNAME_E,
    END OF LTY_STRU_INPUT.
  TYPES:
    BEGIN OF LTY_STRU_LIST,
*     add fields for the selection list here
      ZZNUMBER TYPE ZZNUMBER_E,
      ZZNAME TYPE ZZNAME_E,
    END OF LTY_STRU_LIST.

  DATA: LS_SEARCH_INPUT    TYPE LTY_STRU_INPUT,
```

```
            LT_SELECT_LIST      TYPE STANDARD TABLE OF LTY_STRU_LIST,
            LS_TEXT             TYPE WDR_NAME_VALUE,
            LT_LABEL_TEXTS      TYPE WDR_NAME_VALUE_LIST,
            LT_COLUMN_TEXTS     TYPE WDR_NAME_VALUE_LIST,
            LV_WINDOW_TITLE     TYPE STRING,
            LV_GROUP_HEADER     TYPE STRING,
            LV_TABLE_HEADER     TYPE STRING.

  FIELD-SYMBOLS: <LS_QUERY_PARAMS> TYPE LTY_STRU_INPUT,
                 <LS_SELECTION>       TYPE LTY_STRU_LIST.

  CASE OVS_CALLBACK_OBJECT->PHASE_INDICATOR.

    WHEN IF_WD_OVS=>CO_PHASE_0.   "configuration phase, may be omitted
*   in this phase you have the possibility to define the texts,
*   if you do not want to use the defaults (DDIC-texts)

      LS_TEXT-NAME = 'ZZNUMBER'.    "must match a field name of search
      LS_TEXT-VALUE = '番号'. "wd_assist->get_text( '001' ).
      INSERT LS_TEXT INTO TABLE LT_LABEL_TEXTS.

      LS_TEXT-NAME = 'ZZNAME'.    "must match a field in list structure
      LS_TEXT-VALUE = '姓名'. "wd_assist->get_text( '002' ).
      INSERT LS_TEXT INTO TABLE LT_COLUMN_TEXTS.

*       lv_window_title = wd_assist->get_text( '003' ).
*       lv_group_header = wd_assist->get_text( '004' ).
*       lv_table_header = wd_assist->get_text( '005' ).

      OVS_CALLBACK_OBJECT->SET_CONFIGURATION(
                LABEL_TEXTS    = LT_LABEL_TEXTS
                COLUMN_TEXTS = LT_COLUMN_TEXTS
                GROUP_HEADER = LV_GROUP_HEADER
                WINDOW_TITLE = LV_WINDOW_TITLE
                TABLE_HEADER = LV_TABLE_HEADER
                COL_COUNT      = 2
                ROW_COUNT       = 20 ).

    WHEN IF_WD_OVS=>CO_PHASE_1.   "set search structure and defaults
*   In this phase you can set the structure and default values
*   of the search structure. If this phase is omitted, the search
*   fields will not be displayed, but the selection table is
*   displayed directly.
*   Read values of the original context (not necessary, but you
*   may set these as the defaults). A reference to the context
*   element is available in the callback object.
```

```
          OVS_CALLBACK_OBJECT->CONTEXT_ELEMENT->GET_STATIC_ATTRIBUTES(
              IMPORTING STATIC_ATTRIBUTES = LS_SEARCH_INPUT ).
*         pass the values to the OVS component
          OVS_CALLBACK_OBJECT->SET_INPUT_STRUCTURE(
              INPUT = LS_SEARCH_INPUT ).


        WHEN IF_WD_OVS=>CO_PHASE_2.
*       If phase 1 is implemented, use the field input for the
*       selection of the table.
*       If phase 1 is omitted, use values from your own context.

          IF OVS_CALLBACK_OBJECT->QUERY_PARAMETERS IS NOT BOUND.
******** TODO exception handling
          ENDIF.
          ASSIGN OVS_CALLBACK_OBJECT->QUERY_PARAMETERS->*
                                      TO <LS_QUERY_PARAMS>.
          IF NOT <LS_QUERY_PARAMS> IS ASSIGNED.
******** TODO exception handling
          ENDIF.

*         call business logic for a table of possible values
          SELECT     ZZNUMBER    ZZNAME   FROM   ZMNUMBER_T
                 INTO TABLE LT_SELECT_LIST .
          OVS_CALLBACK_OBJECT->SET_OUTPUT_TABLE( OUTPUT = LT_SELECT_LIST ).

        WHEN IF_WD_OVS=>CO_PHASE_3.
*       apply result

          IF OVS_CALLBACK_OBJECT->SELECTION IS NOT BOUND.
******** TODO exception handling
          ENDIF.

          ASSIGN OVS_CALLBACK_OBJECT->SELECTION->* TO <LS_SELECTION>.
          IF <LS_SELECTION> IS ASSIGNED.
            OVS_CALLBACK_OBJECT->CONTEXT_ELEMENT->SET_ATTRIBUTE(
                                       NAME    = 'ZZNUMBER'
                                       VALUE = <LS_SELECTION>-ZZNUMBER ).
            OVS_CALLBACK_OBJECT->CONTEXT_ELEMENT->SET_ATTRIBUTE(
                                      NAME    = 'ZZNAME'
                                      VALUE = <LS_SELECTION>-ZZNAME ).
*           or
*             ovs_callback_object->context_element->set_static_attributes(
*                                        static_attributes = <ls_selection> ).
```

```
        ENDIF.
    ENDCASE.

  ENDMETHOD.
```

6）在 ABAP 工作台中的 Web Dynpro 应用编辑界面创建 Web Dynpro Application，如图 7-82 所示。

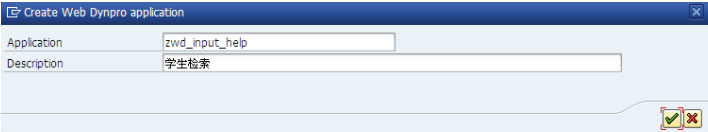


图 7-82

7）在 ABAP 工作台中，测试结果如图 7-83 所示。

学生信息

学生番号: 00001

检索帮助: 00001

OVS: 00001

Free: 00001

姓名: 郑喜定

图 7-83

自动弹出的输入帮助，如图 7-84 所示。

学生番号: allgemeine Werteliste

Pers鲢liche Werteliste　Einstellungen...

Suchkriterien ausblenden

学生学号:

姓名:

Anzahl der Eintr鋑e in der Werteliste einschr鋘ken auf 500

Suche starten　Zur點ksetzen

Zur pers鲢lichen Werteliste hinzuf黦en

学...	姓名
00001	郑喜定
00002	魏喜奎
00003	金英
00004	洪喜官
00005	朴珍
00006	王喜凤
00007	刘喜仁
00008	白喜医
00009	赵喜庆

Weitere Suchhilfen: 学生查询-学号

确定　取消

图 7-84

检索帮助，如图 7-85 所示。

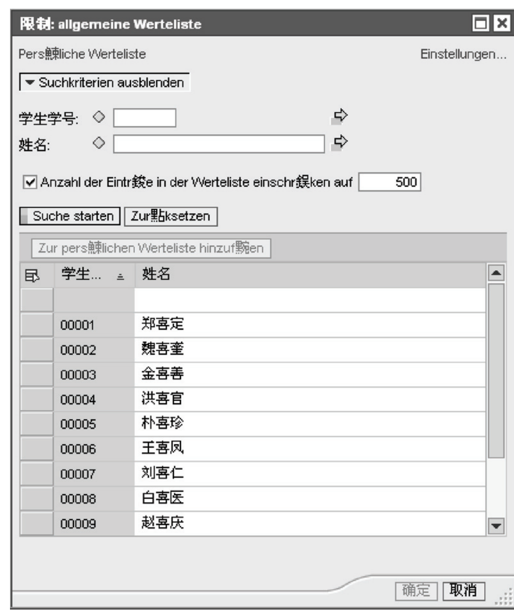


图 7-85

OVS 帮助，如图 7-86 所示。



图 7-86

自定义检索帮助，如图 7-87 所示。



图 7-87

7.5 ABAP Web Dynpro 应用程序的帮助文本

ABAP Web Dynpro 可使用多种方式显示帮助文本。

帮助文本可以与 UI 元素相关。在这里，工具提示也是常用功能。用户将光标置于相关的 UI 元素上，即可使用该类语义帮助显示短帮助文本。说明文本可用来在相关字段下方直接显示较长信息文本。最后，还可以在 ABAP Web Dynpro Context 中使用标准字典 F1 帮助。

如果必须显示与特定 UI 元素不相关的一般帮助，则可使用 EXPLANATION 元素。另外，可以可在所谓的“帮助中心”加载并显示知识仓库中所定义的文档。

Web Dynpro 应用程序（各种 UI 元素视图）中最终用户可以使用各种方式显示简短的帮助文本，也可以在一个完整的应用程序或窗体中显示较长的文本文档。

（1）UI 元素相关帮助文本

可采用以下技术将帮助文本关联到指定 UI 元素。

- Explanation 元素。
- Explanation 属性。
- 典型的 F1 帮助。

（2）应用程序或窗体（帮助中心）相关帮助文本

对于一个 Web Dynpro 应用程序或窗体，可以存储一个帮助文本（信息对象）。该对象信息可以在知识库（Knowledge Warehouse）中创建。要做到这一点，需转到应用程序或窗体的属性配置页面。

- Help Menu Text 处指定帮助窗体的标题。
- Help Link 处指定帮助链接所需知识仓库系统的信息对象，可以使用输入字段的右侧标示来创建或者修改链接。为此需要定义到知识仓库系统的 RFC 连接 AIO_FOR_HELP_LINKS。

注：当程序员选择知识仓库中的相关信息对象时，操作的是相关生产系统和相关区域的知识仓库 Context，如文档。当选择知识仓库中的信息对象时，有以下选项：

- 选择一个主题，继而选择相关的结构。
- 选题一个主题但不选择结构（不选择任何结构并关闭结构选择对话框）。

需要注意的是，知识库中的主题和结构必须被释放。

用户还可以为 Web Dynpro ABAP 程序和窗体定义不是一个单一的而是多个帮助链接。这些额外的帮助链接在运行时通过帮助中心的 Helpful Links 显示。程序员可以使用程序 WDY_AUTHORING_HELP_LINKS 创建链接。

帮助可以通过在运行时选择 F1 或标题栏中的“帮助”按钮被调用。然后该帮助中心和该应用程序的窗体被显示。对快速帮助的解释、知识仓库文档、SAP 库的链接，以及定义为应用程序窗体的帮助链接都存储在帮助中心。

程序员可以在应用程序中创建一个链接，这样应用程序本身就可以触发帮助中心。进行如下编程：

- 前期准备。
- 插入一个 LinkToAction 元素到视图。
- 插入非验证操作。
- 定义下列源代码：

```
data: l_api_componentcontroller type ref to if_wd_component.
data l_appl type ref to if_wd_application.

l_api_componentcontroller = wd_comp_controller->wd_get_api( ).
l_appl = l_api_componentcontroller->get_application( ).
l_appl->open_help_center( ).
```

示例：

此处由 Help Center 连接到 SAP 的官方网站，如图 7-88 所示。

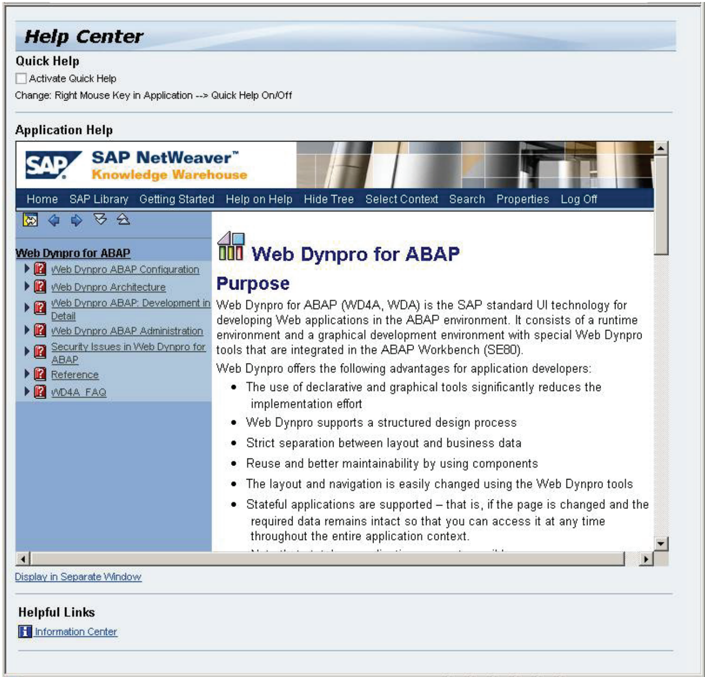


图 7-88

（3）动态重写应用程序帮助链接

如果用户不想 Web Dynpro 应用定义的帮助链接显示在帮助中心，而是显示链接到其他“帮助”主题的帮助连接，就可以重写应用程序帮助动态链接。要做到这一点，需使用接口 IF_WD_RR_APPLICATION 的方法 SET_HELP_LINK 并指定 APPLICATION_INFO。

示例：

```
method onactionchange_appl_helplink .
  data api_main2          type ref to if_wd_view_controller.
  data component          type ref to if_wd_component.
  data application        type ref to if_wd_application.
  data application_info type ref to if_wd_rr_application.
  api_main2 = wd_this->wd_get_api( ).
  component = api_main2->get_component( ).
  application = component->get_application( ).
  application_info = application->get_application_info( ).

application_info->set_help_link( 'HELPTYPE=SAPHELP&_LOIO=452C8201E9EF48D882BD10244D085661&
_CLASS=BIC_OOO' ).
endmethod.
```

注：可以在系统组件 WDR_TEST_HELP 中找到示例代码。

7.5.1 Explanation UI 元素

工具提示可以在黄色框中显示短文本，最大长度为 255。如果鼠标移动到分配有工具提示的字段上，就会出现该框。如果鼠标从该字段占用区域移开，则该工具提示将会消失。

通过设定相应属性的工具提示，可将工具提示分配到 UI 元素。该文本可在属性字段中输入，来源于在线文本资源库（OTR）中定义的别名文本可以是 Context 属性的内容，或者是与数据元素相关的文本之一。

根据在视图的 UI 元素层次结构中所处位置，工具提示可由其他的 UI 元素继承。

程序员可以在视图 View 上使用 Explanation 元素来显示解释性文本。Explanation 元素包含一个文本，可以显示在屏幕上，并且该解释应该包含屏幕或部分屏幕。

在视图设计器中，可以创建文本属性的解释文本。文本可以是手工输入（最大 255 个字符），也可以是一个 OTR 别名文本，可以使用符号$ OTR：Aliasname 访问，还可以写一个脚本，允许文本超过 255 个字符，可以是各种格式。要做到这一点使用方法 WDDOMODIFYVIEW。

示例：

```
method wddomodifyview.
  data lines              type tlinetab.
  data header             type thead.
  data text               type string.
  data wad_formatted_text type ref to cl_wd_formatted_text.
```

```
      data ui_explanation          type ref to cl_wd_explanation.
      call function 'DOCU_GET'
        exporting
          id                        = 'TX'
          langu                     = sy-langu
          object                    = 'WDR_TEST_HELP_EXP1'
        importing
          head                      = header
        tables
          line                      = lines
        exceptions
          no_docu_on_screen = 1
          no_docu_self_def    = 2
          no_docu_temp          = 3
          ret_code                  = 4
          others                    = 5.
      if sy-subrc <> 0.
        text = space.
      endif.
      wad_formatted_text = cl_wd_formatted_text=>create_from_sapscript( sapscript_head    = header
                                                                  apscript_lines = lines
                                                                  type = cl_wd_formatted_text=>e_type-wad
                      ).
      if wad_formatted_text is bound.
        text = wad_formatted_text->m_xml_text.
      else.
        text = ''.
      endif.
      ui_explanation ?= view->get_element( 'ON_SCREEN_EXPL1' ).
      ui_explanation->set_text( text ).
      wd_this->m_view = view.
    endmethod.
```

当设计强调某个 UI 元素时，一个说明文本高亮显示，与其他的用户界面文本不同，文本为蓝色并带有下画线，开关标示 Hide Quick Help 也被显示。

示例：

Explanation UI 元素，如图 7-89 所示。

图 7-89

与帮助中心连接的复杂的例子，如图 7-90 所示。

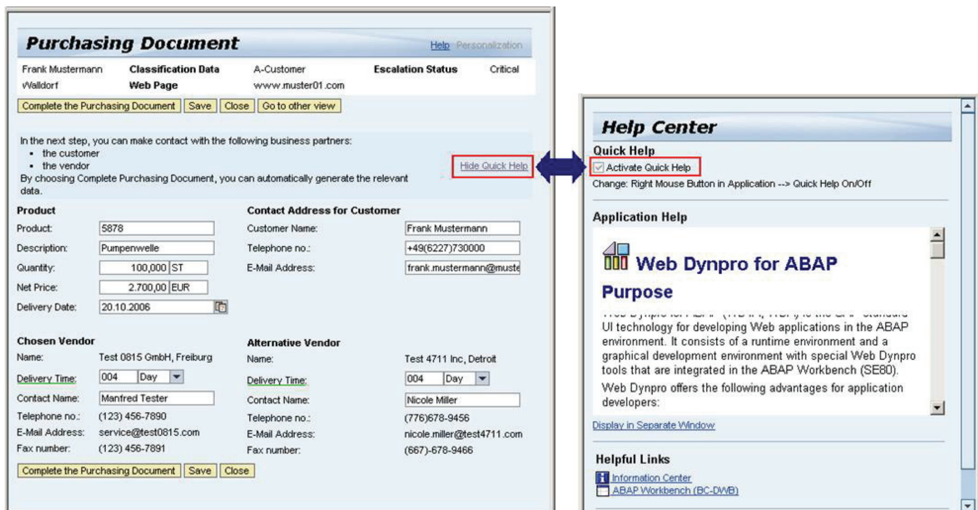


图 7-90

从浏览器中通过 Context 菜单可以激活/停用解释文本的显示（Quick Help On/Off），如图 7-91 所示。

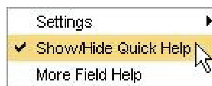


图 7-91

在 Context 菜单中，更多字段帮助（More Field Help）选项用于显示技术信息以及更全面的数据元素的文档。如果没有连接到 ABAP，则只显示技术信息。

7.5.2 Explanation 属性

还可以为不同的 UI 元素设置 Explanation 元素。该属性是否显示取决于帮助模式是否打开或关闭。根据帮助模式有以下不同功能：

（1）Help mode = on

如果对特定的 UI 元素设置了 Explanation 属性，则该 UI 元素在屏幕上显示为绿色并带有下画线。如果用户单击分配有说明文本的 UI 元素或者将光标移动到绿色的 UI 元素标记，则显示该属性的内容。另外，这是相对独立的，任何 UI 元素都可用的工具提示。在这种情况下，属性实际上包含一个 F1 帮助。

示例：

Explanation 元素作为帮助提示，如图 7-92 所示。

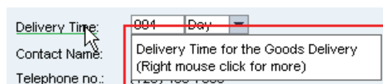


图 7-92

注：提示性内容仅在读取模式下由屏幕阅读器读取。

（2）Help mode = off

说明内容不显示。系统不提供主要的帮助。

在门户网站中的个人设置或相关联的 URL 参数中，可以激活帮助模式。

在视图设计器中，可以为该属性的值创建输入字段中的 Explanation 属性的文本。文本可以是手工输入（最大为 255 个字符），也可以是一个 OTR 别名的文本。该文本使用符号$ OTR：Aliasname 可以访问。也可以使用 F4 帮助指定该字段的数据元素的文档或它的一部分作为帮助文本显示在屏幕上。这将是数据元素的数据元素文档，需要用户在分配选项中绑定属性中的主属性（Primary Property）。可以选择下列显示变量为数据元素文档：

ABAP 字典中的短文本

- 数据元素的长文本中关键块&DEFINITION&的定义。
- 数据元素的长文本中关键块 &USE&的定义。
- 完整的数据元文件（长文本）的链接。

当鼠标指针移动到相应的 UI 元素时，帮助文本显示在屏幕上。

注：如果 UI 元素的主属性（即输入字段的值属性）绑定到 Context 属性，则文本还可源于与该 Context 属性相关的数据元素。开发人员可确定是否显示数据元素中存储的短文本、关键块&DEFINITION& 的内容或关键块 &USE&的内容，这些关键块来自于数据元素文档。

各种帮助选项指定如图 7-93 所示。

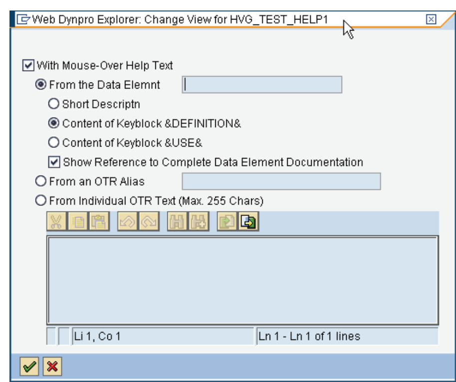


图 7-93

要显示数据元素的文档，相应的 UI 元素必须有主属性（Primary Property），并且要在 ABAP 字典中绑定一个数据元素到数据元素文档。如果没有这样的数据元素，则弹出的数据元素文件选项是灰色的。在这种情况下，显示数据元素文档是不可能的，因为没有设定主属性。

示例：

以不同方式显示部分数据元素文档的例子，如图 7-94 所示。

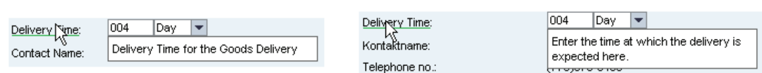


图 7-94

数据元素 WDR_TEST_HELP_DELIVERY_TIME，左边为文本定义（&DEFINITION&），

右边为文本使用（&USE&），如图 7-95 所示。

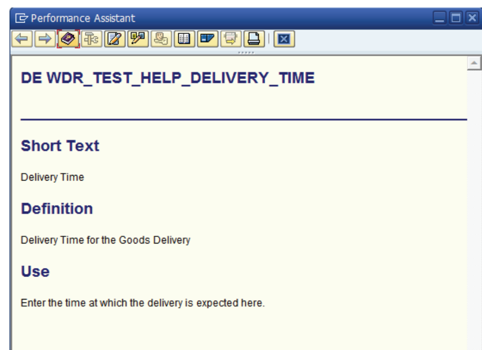


图 7-95

可以为下列 UI 元素设置 Explanation 属性：AbstractTableColumn、Button、CheckBox、DropDownByIndex、DropDownByKey、FileUpload、InputField、ItemListBox、RadioButton、TextEdit、TriStateCheckBox。

（3）关闭 F1 帮助

默认情况下，当帮助模式被激活时，将显示数据元素文档。但是也可以关闭数据元素文档的显示并为个别字段定义例外。应用参数 WDHIDEMOREFIELDHELPASDEFAULT 为此提供相应功能。

如果该参数设置为 X，并且显示说明（Display Note）为所有数据元素文档（Full Data Element Documentation）选项是激活的，并且帮助文本鼠标选项（Help Text for Mouse-Over）和数据元素（From the Data Element）的某一属性关联 UI 元素的 Explanation 属性被设定，在文本信息后的括号中将显示单击鼠标右键更多（Right Mouse Button for More）的字样。使用鼠标右键单击更多字段将有助（More Field Help）选项显示相关的数据元素文档。

7.5.3 典型 F1 帮助

可以使用 UI 元素的 F1 帮助功能（如同 SAP GUI）。要做到这一点，相关的数据元素的文件必须已经存在于系统中，内容元素必须将 F1 帮助连接到一个 ABAP 字典字段。

用户可以使用组合键〈Ctrl + F1〉，或者通过在 Context 菜单中选择更多的字段帮助选项，然后显示系统的文档。

示例：在 Web Dynpro ABAP 应用数据元素文档的例子，如图 7-96 所示。

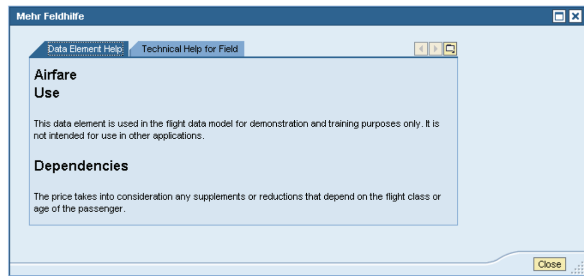


图 7-96

7.6 消息 Messages

在 Web Dynpro ABAP 中，程序员可以创建和显示消息。该信息包含 Web Dynpro 应用程序与最终用户交互的重要信息。消息依赖于文本语言。例如，如果一个应用程序运行时发生错误，或者用户输入的数据格式错误，则相应错误信息显示在屏幕上。

注： NW7.0 SPS11 版本的消息和消息区都被赋予了新的设计，提高了用户的指导，是类似于旧的设计。新的设计是默认设置。不过，也可以切换到旧的设计。

示例：

用户检查，如图 7-97 所示。

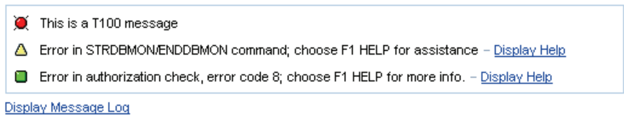


图 7-97

对于这些用户信息，如信息（Information）、错误消息（error）和警告（warnings），ABAP 应用程序服务器在 Web Dynpro 运行时提供了运行时服务。消息组件是 Web Dynpro 应用程序的一部分，如果需要可在 Web Dynpro 应用程序的设置中配置。有以下 3 种可能处理消息的设置：

1）如果需要，显示消息组件，如图 7-98 所示。

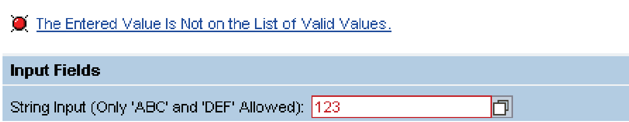


图 7-98

如果消息存在，则它们被显示出来。

示例：

2）始终显示消息组件，如图 7-99 所示。

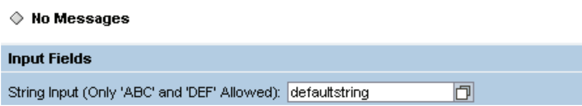


图 7-99

即使没有消息，该消息仍然显示组件在顶视图中。

示例：

3）无消息组件显示消息，如图 7-100 所示。

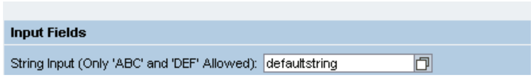


图 7-100

隐藏消息组件。信息在初期显示。有了这种类型的信息处理只有一个消息显示，没有消息日志。

此选项仅适用于较小的测试应用程序。

用户消息显示在状态栏的链接中。在错误报告中，用户可以使用链接导航到可用于删除错误的 UI 元素。输入的焦点是自动传输，从而大大增加了信息效率。它也可以在屏幕上显示多个消息表。

程序员可以指定消息显示在具体的视图和窗体中，这样消息就会实时地显示在主窗体内，而不是显示在弹出的窗体内。要做到这一点，可以在消息管理器的方法中使用可选参数 VIEW 指定有关的视图或窗体的名称。

消息要定义在一个 Web Dynpro 组件的级别。ABAP 开发工具包括一个消息维护的图形工具。

7.6.1 消息弹出

弹出消息显示是标准配置的需求，无论在应用程序中是如何设置的。

弹出窗体可以配置一个标示用于显示：

- 所有的信息，如截至目前（标准设置）。
- 只有那些属于自己的窗体，以及所有无窗体特定的消息。
- 不显示任何消息。

（1）基本功能

1）显示，如图 7-101 所示

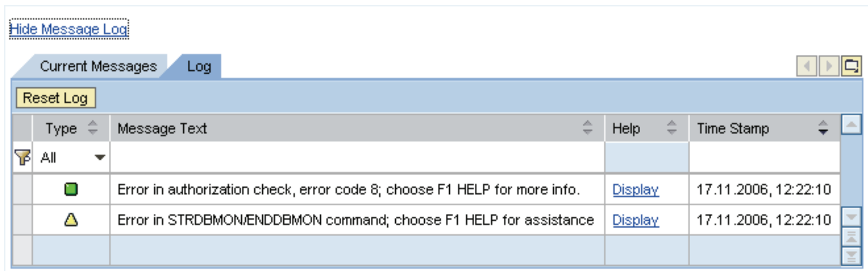


图 7-101

- 用户最后交互相关的当前消息。
- 最后的信息，在消息日志列表的上端，具有最高的权重。
- 一个消息的计数器和所有消息的计数器。
- 标示每个消息权重的符号。
- 所有消息的时间标示。

2）要导航到消息，用户需要单击消息关联链接。

3）消息在标签条中按照下列条件排序。

- 权重。
- 文本。
- 事件标示。

4）通过过滤标准可输入：

- 权重。
- 文本。
- 事件标示。

5）重置整个消息记录，即删除当前消息日志。

6）显示和隐藏消息日志。

- 隐藏。只有头部的最后一条消息和消息计数器仍然显示。
- 显示。当前信息表视图和消息总数都被显示。

（2）集成

在应用程序开发中，UI 元素 MessageArea [外部]提供了消息显示在屏幕上的定位。该过程是将消息集成到消息日志中。

示例：系统组件 WDR_MESSAGE_AREA 和 WDR_TEST_INPUT。

7.6.2 消息日志中的消息集成

消息集成是使用消息管理器在一个组件中将其集成到消息日志（接口 IF_WD_MESSAGE_MANAGER）。消息管理器通过代码向导（外部）被集成到 Web Dynpro。以下方法用于触发消息。

1）CLEAR_MESSAGES：删除所有消息。

2）IS_EMPTY：查询消息是否存在。

3）REPORT_ATTRIBUTE_ERROR_MESSAGE：报告一个 Context 属性的 Web Dynpro 例外。

4）REPORT_ATTRIBUTE_EXCEPTION：报告一个 Context 属性的 Web Dynpro 例外。

5）REPORT_ATTRIBUTE_T100_MESSAGE：报告一个 Context 属性的 Web Dynpro 例外。

6）REPORT_ERROR_MESSAGE：报告一个带有可选参数的 Web Dynpro 消息。

7）REPORT_EXCEPTION：报告一个 Web Dynpro 例外（它可以返回）。

8）REPORT_FATAL_ERROR_MESSAGE：报告一个致命的带有可选参数的 Web Dynpro 消息。

9）REPORT_FATAL_EXCEPTION：报告一个致命的 Web Dynpro 消息。

10）REPORT_SUCCESS：报告成功消息。

11）REPORT_T100_MESSAGE：报告一个消息，使用 T100 条目。

12）REPORT_WARNING：报告警告。

注：消息管理的 RAISE 方法是过时的。因为这个原因，程序员不应该使用它。有关的方法，可以使用可选参数视图输入要显示消息的窗体或视图的名称。

示例：

```
* 触发一个  T100  消息
* get message manager
data: l_current_controller type ref to if_wd_controller,
```

```
l_message_manager type ref to if_wd_message_manager.
l_current_controller ?= wd_this->wd_get_api( ).
call method l_current_controller->get_message_manager
  receiving
    message_manager = l_message_manager.
* report message
call method l_message_manager->report_t100_message
exporting
msgid =’bc’
msgno =’005’
msgty =’e’
* P1 =
* P2 =
* P3 =
* P4 =
.
```

7.6.3 配置消息

可以配置消息是如何在应用程序的视图或窗体级别显示的。要做到这一点，需在方法 WDDOINIT 中加上下面的源代码，具体参数见表 7-1。

```
data:
    l_api_mycomp type ref to if_wd_window_controller,
    l_wd_message_area type ref to if_wd_message_area.

l_api_mycomp ?= wd_this->wd_get_api( ).
l_wd_message_area = l_api_mycomp->get_message_area( ).
l_wd_message_area->set_display_attributes( i_for_all_instances = ' '
                                           i_msg_lines_visible = '0'
                                           i_use_toggle_area     = '   '
                                           i_show_only_current = ' ' ).
```

表 7-1

设　置	说　明
i_use_toggle_area = ' '	消息区（默认设置）的新设计
i_use_toggle_area = 'X'	消息区的旧设计（可切换新设计）
i_msg_lines_visible = 0	所有的消息都立即可见（默认设置）
i_msg_lines_visible = <x>	x > 0。只有 X 的消息立即可见，其余部分滚动时可见
i_show_only_current = 'X'	与参数一并设置为 i_use_toggle_area ='，这意味着任何消息日志可以显示（默认设置）
i_show_only_current = ' '	与参数一并设置为 i_use_toggle_area ='，这意味着一个链接连接到消息日志可以显示

注：程序员可以在程序代码中删除消息记录。要做到这一点，需使用方法 RESET_MESSAGES。

7.6.4 重用组建和 MessageArea 元素

如果程序员使用可重用组件，则需确保组件中包含可重用的元素。可重用组件要使用在多个应用中，不能使用 MessageArea UI 元素，由于每个窗体最多可以只包含一个可见的消息区。

举一个例子，组件可以是独立的和可重用的。创建了两个视图，第一个视图只包含不会被重复使用的 UI 元素，如 Header、MessageArea 和 ViewContainer。第二个视图只包含可嵌入和重用的核心应用程序。在组件中创建两个窗体。第一个是运行独立应用程序的默认窗体。在这里，嵌入第一视图。把第二个视图放在 UI 元素 ViewContainer 中。第二个窗体作为界面视图，用来嵌入到第二个视图。

7.6.5 编辑 Web Dynpro 应用程序

下面的例子通过自定义的组件辅助类完成信息操作。

1）在 ABAP 工作台中的编辑界面创建 Web Dynpro 组件。

选择“Web Dynpro Comp./Intf.”，在相应的文本框中输入程序的名称，如图 7-102 所示。

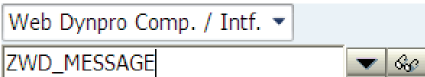


图 7-102

单击图 7-102 中的按钮，创建组件如图 7-103 所示。

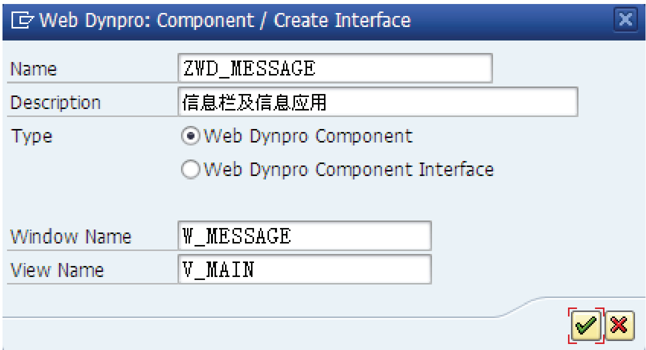


图 7-103

2）ABAP 工作台中的编辑界面创建 Web Dynpro 组件辅助类，如图 7-104 所示。

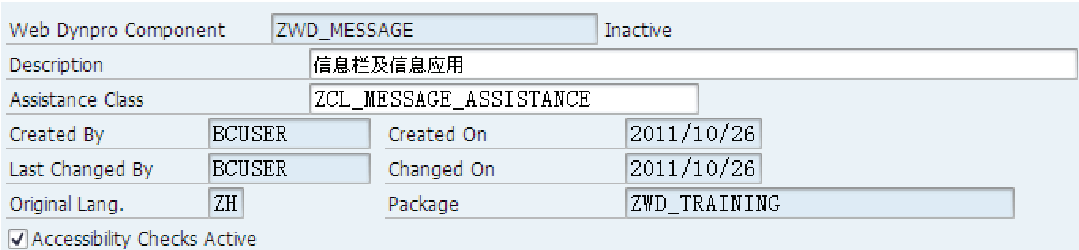


图 7-104

编辑辅助类文本符号，如图 7-105 所示。

Class ZCL_MESSAGE_ASSISTANCE Inactive

Text Symbols

Sym	Text	dLen	mLen
001	数字需大于!	12	18
002	数字在100以内。	15	23
003	数字刚刚好!	12	18
004	数字在334到666之间。	20	20
005	数字在667到900之间。	20	20
006	数字 &P1 大于900。	18	18
		0	

图 7-105

编辑辅助类属性如图 7-106 所示。

Klassenschnittstelle ZCL_MESSAGE_ASSISTANCE 已实现 /

Eigenschaften Interfaces Friends Attribute Methoden Ereignisse Typen 别名

筛选器

属性	级别	可...	只读	Typisieru...	参考打印		说明	初始值
TEXT_NUM	Constant	Public					文本文字编号	
CONTEXT_ATTR	Constant	Public		Type	STRING		CONTEXT属性	'NUMBERS'
				Type				

图 7-106

单击按钮，编辑常量 TEXT_NUM 如下：

```
CONSTANTS:
  BEGIN OF TEXT_NUM,
  N001 TYPE WDR_TEXT_KEY VALUE '001',
  N002 TYPE WDR_TEXT_KEY VALUE '002',
  N003 TYPE WDR_TEXT_KEY VALUE '003',
  N004 TYPE WDR_TEXT_KEY VALUE '004',
  N005 TYPE WDR_TEXT_KEY VALUE '005',
  N006 TYPE WDR_TEXT_KEY VALUE '006',
  END OF    TEXT_NUM .
```

3）在 ABAP 工作台中的编辑界面编辑 OTR 文本，如图 7-107 所示。

ZWD_TRAINING		
数字在334到666之间。	ZWD_TRAINING/EXCEPTION_WANGMING	BCUSER
数字在667到900之间。	ZWD_TRAINING/EXCEPTION_ERROR	BCUSER

图 7-107

具体 Text 文本编辑如图 7-108 和图 7-109 所示。

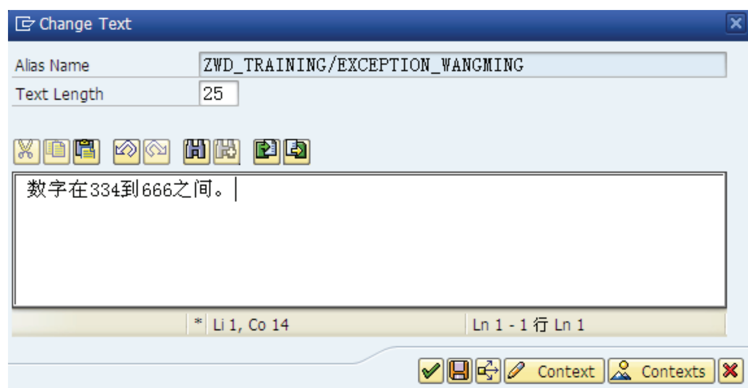


图 7-108

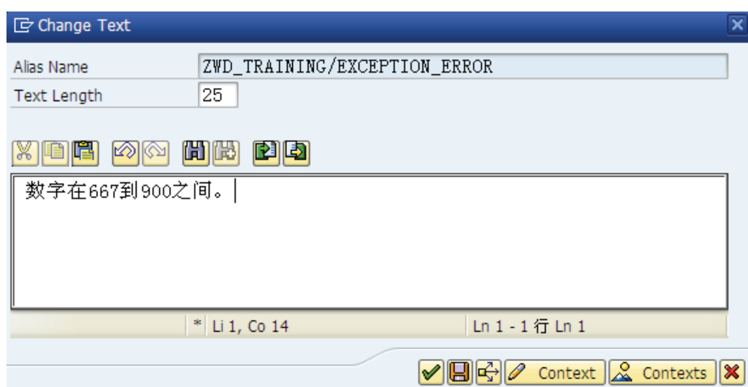


图 7-109

编辑 Context 节点如图 7-110 所示。

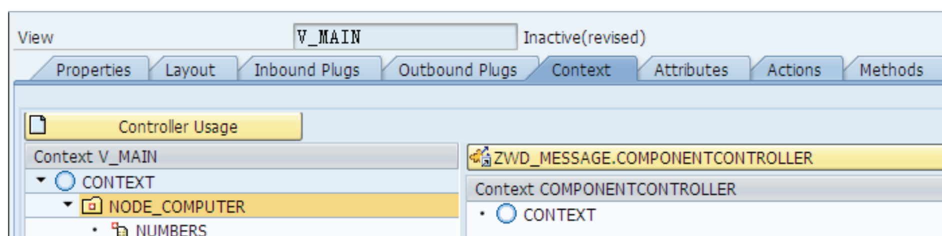


图 7-110

选择“Layout”选项卡，如图 7-111 所示。

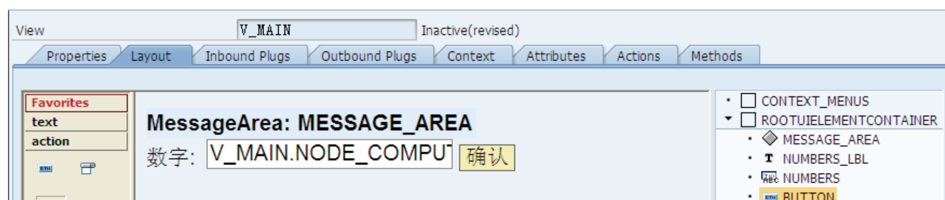


图 7-111

为按钮“确认”编辑 Action 如图 7-112 所示。

Create Action

Component	ZWD_MESSAGE
View	V_MAIN
Action	CONFIRM
Description	确认

Select an outbound plug or enter an outbound plug for leaving the view by selecting the pushbutton

Outbound Plug

图 7-112

编辑方法如下：

```
METHOD ONACTIONCONFIRM .
  DATA LO_ND_NODE_COMPUTER TYPE REF TO IF_WD_CONTEXT_NODE.
  DATA LO_EL_NODE_COMPUTER TYPE REF TO IF_WD_CONTEXT_ELEMENT.
  DATA LS_NODE_COMPUTER TYPE WD_THIS->ELEMENT_NODE_COMPUTER.

*   navigate from <CONTEXT> to <NODE_COMPUTER> via lead selection
  LO_ND_NODE_COMPUTER = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->
WDCTX_NODE_COMPUTER ).
*   get element via lead selection
  LO_EL_NODE_COMPUTER = LO_ND_NODE_COMPUTER->GET_ELEMENT( ).
*   get all declared attributes
  LO_EL_NODE_COMPUTER->GET_STATIC_ATTRIBUTES(
    IMPORTING
      STATIC_ATTRIBUTES = LS_NODE_COMPUTER ).
  IF LS_NODE_COMPUTER-NUMBERS <= 1 .
    WD_THIS->SET_ERROR(
    ).

  ELSEIF LS_NODE_COMPUTER-NUMBERS>=2 AND LS_NODE_COMPUTER-NUMBERS <=   100.
    WD_THIS->SET_WARNING(
    ).
  ELSEIF LS_NODE_COMPUTER-NUMBERS >= 101 AND LS_NODE_COMPUTER-NUMBERS
<=   333 .
    WD_THIS->SET_SUCCESS(
    ).
  ELSEIF LS_NODE_COMPUTER-NUMBERS >= 334 AND LS_NODE_COMPUTER-NUMBERS
<=   666 .
    WD_THIS->SET_EXCEPTION_WARNING(
```

```
            ).
        ELSEIF LS_NODE_COMPUTER-NUMBERS >= 667 AND LS_NODE_COMPUTER-NUMBERS
<= 900 .
            WD_THIS->SET_EXCEPTION_ERROR(
            ).
        ELSE.
            WD_THIS->SET_ERROR_PARM(
            ).

        ENDIF.

    ENDMETHOD.
```

选择“Methods”选项卡，编辑消息抛出方法体，如图 7-113 所示。

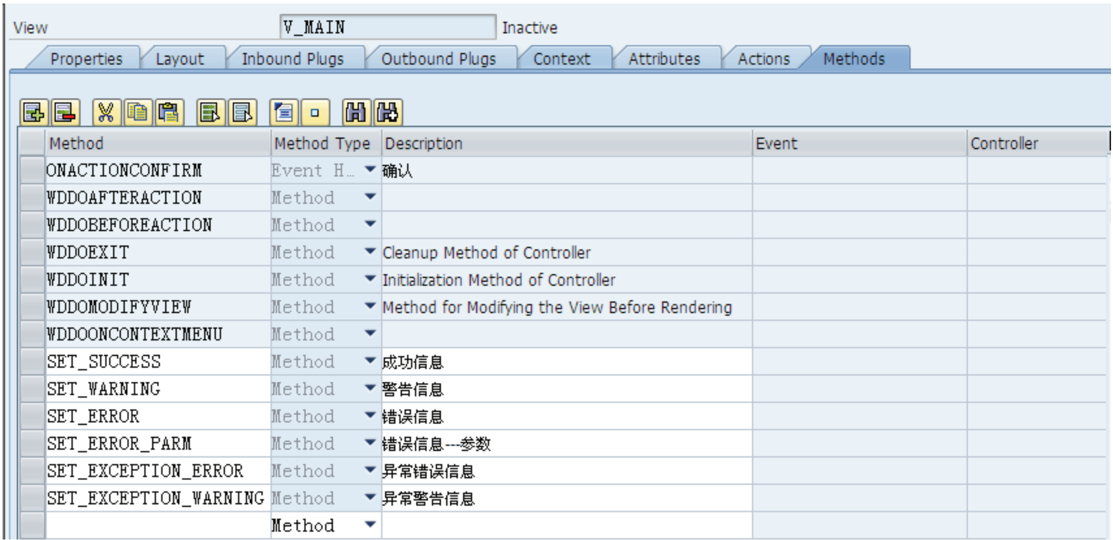


图 7-113

SET_SUCCESS 方法：

```
    METHOD SET_SUCCESS .
        DATA: L_TEXT TYPE STRING.
    *   取得文字
        L_TEXT =
        WD_ASSIST->IF_WD_COMPONENT_ASSISTANCE~GET_TEXT(WD_ASSIST->TEXT_NUM-N003).
    *     get message manager
        DATA: L_CURRENT_CONTROLLER TYPE REF TO IF_WD_CONTROLLER,
                    L_MESSAGE_MANAGER       TYPE REF TO IF_WD_MESSAGE_MANAGER.
        L_CURRENT_CONTROLLER ?= WD_THIS->WD_GET_API( ).
        L_MESSAGE_MANAGER = L_CURRENT_CONTROLLER->GET_MESSAGE_MANAGER( ).
    *     report message
        L_MESSAGE_MANAGER->REPORT_SUCCESS(
```

```
    MESSAGE_TEXT   = L_TEXT
*         PARAMS              =
*         MSG_USER_DATA =
        ).
ENDMETHOD.
```

SET_ WARNING 方法：

```
METHOD SET_SUCCESS .
  DATA: L_TEXT TYPE STRING.
*   取得文字
  L_TEXT = WD_ASSIST->IF_WD_COMPONENT_ASSISTANCE~GET_TEXT( WD_ASSIST->TEXT_
NUM-N003 ).
*    get message manager
  DATA: L_CURRENT_CONTROLLER TYPE REF TO IF_WD_CONTROLLER,
             L_MESSAGE_MANAGER       TYPE REF TO IF_WD_MESSAGE_MANAGER.
  L_CURRENT_CONTROLLER ?= WD_THIS->WD_GET_API( ).
  L_MESSAGE_MANAGER = L_CURRENT_CONTROLLER->GET_MESSAGE_MANAGER( ).
*    report message
  L_MESSAGE_MANAGER->REPORT_SUCCESS(
         MESSAGE_TEXT   = L_TEXT
*         PARAMS              =
*         MSG_USER_DATA =
            ).
ENDMETHOD.
```

SET_ ERROR 方法：

```
METHOD SET_ERROR .

  DATA: L_TEXT TYPE STRING.
*    get message manager
  L_TEXT = WD_ASSIST->IF_WD_COMPONENT_ASSISTANCE~GET_TEXT( WD_ASSIST->
TEXT_NUM-N001 ).
  DATA: L_CURRENT_CONTROLLER TYPE REF TO IF_WD_CONTROLLER,
             L_MESSAGE_MANAGER       TYPE REF TO IF_WD_MESSAGE_MANAGER.
  L_CURRENT_CONTROLLER ?= WD_THIS->WD_GET_API( ).
  L_MESSAGE_MANAGER = L_CURRENT_CONTROLLER->GET_MESSAGE_MANAGER( ).
*    report message
  L_MESSAGE_MANAGER->REPORT_ERROR_MESSAGE(
        MESSAGE_TEXT   = L_TEXT
*         PARAMS              =
*         MSG_USER_DATA =
).
ENDMETHOD.
```

SET_ ERROR_PARM 方法：

```
METHOD SET_ERROR_PARM .
  DATA: L_TEXT TYPE STRING,
        L_VALUE TYPE I,
        L_PARAMS TYPE WDR_NAME_VALUE_LIST,
        L_PARAM TYPE WDR_NAME_VALUE.
  DATA LO_ND_NODE_COMPUTER TYPE REF TO IF_WD_CONTEXT_NODE.
  DATA LO_EL_NODE_COMPUTER TYPE REF TO IF_WD_CONTEXT_ELEMENT.
  DATA LS_NODE_COMPUTER TYPE WD_THIS->ELEMENT_NODE_COMPUTER.
*   navigate from <CONTEXT> to <NODE_COMPUTER> via lead selection
  LO_ND_NODE_COMPUTER = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->
WDCTX_NODE_COMPUTER ).
*   get element via lead selection
  LO_EL_NODE_COMPUTER = LO_ND_NODE_COMPUTER->GET_ELEMENT( ).
*   get single attribute
  LO_EL_NODE_COMPUTER->GET_ATTRIBUTE(
    EXPORTING
      NAME =   WD_ASSIST->CONTEXT_ATTR
    IMPORTING
      VALUE = L_VALUE   ).

  L_TEXT = WD_ASSIST->IF_WD_COMPONENT_ASSISTANCE~GET_TEXT( WD_ASSIST->
TEXT_NUM-N006 ).
  L_PARAM-NAME = 'P1'.
  L_PARAM-VALUE = L_VALUE.
  APPEND L_PARAM TO L_PARAMS.
*   get message manager
  DATA: L_CURRENT_CONTROLLER TYPE REF TO IF_WD_CONTROLLER,
        L_MESSAGE_MANAGER      TYPE REF TO IF_WD_MESSAGE_MANAGER.
  L_CURRENT_CONTROLLER ?= WD_THIS->WD_GET_API( ).
  L_MESSAGE_MANAGER = L_CURRENT_CONTROLLER->GET_MESSAGE_MANAGER( ).
*   report message
  L_MESSAGE_MANAGER->REPORT_ERROR_MESSAGE(
      MESSAGE_TEXT   = L_TEXT
      PARAMS         = L_PARAMS
*        MSG_USER_DATA =
        ).

ENDMETHOD.
```

SET_ EXCEPTION_ERROR 方法：

```
METHOD SET_EXCEPTION_ERROR .
```

```
      DATA: L_EXC TYPE REF TO CX_WDR_DEMO_MESSAGES.
      CREATE OBJECT L_EXC.
    *   get message manager
      DATA: L_CURRENT_CONTROLLER TYPE REF TO IF_WD_CONTROLLER,
            L_MESSAGE_MANAGER    TYPE REF TO IF_WD_MESSAGE_MANAGER.

      L_CURRENT_CONTROLLER ?= WD_THIS->WD_GET_API( ).
      L_MESSAGE_MANAGER = L_CURRENT_CONTROLLER->GET_MESSAGE_MANAGER( ).

    *   report message
      L_MESSAGE_MANAGER->REPORT_EXCEPTION(
          MESSAGE_OBJECT = L_EXC
    *       MSG_USER_DATA    =
          MESSAGE_TYPE     = L_MESSAGE_MANAGER->CO_TYPE_ERROR
             ).

    ENDMETHOD.
```

SET_ EXCEPTION_ WARNING 方法：

```
    METHOD SET_EXCEPTION_WARNING .
      DATA: L_EXC       TYPE REF TO CX_WDR_DEMO_MESSAGES,
            L_VALUE     TYPE I,
            L_ELEMENT TYPE REF TO IF_WD_CONTEXT_ELEMENT.
      DATA LO_ND_NODE_COMPUTER TYPE REF TO IF_WD_CONTEXT_NODE.
      DATA LO_EL_NODE_COMPUTER TYPE REF TO IF_WD_CONTEXT_ELEMENT.
      DATA LS_NODE_COMPUTER TYPE WD_THIS->ELEMENT_NODE_COMPUTER.
    *   navigate from <CONTEXT> to <NODE_COMPUTER> via lead selection
      LO_ND_NODE_COMPUTER = WD_CONTEXT->GET_CHILD_NODE( NAME = WD_THIS->
WDCTX_NODE_COMPUTER ).
    *   get element via lead selection
      LO_EL_NODE_COMPUTER = LO_ND_NODE_COMPUTER->GET_ELEMENT( ).

      TRY.
    *   get message manager
        DATA: L_CURRENT_CONTROLLER TYPE REF TO IF_WD_CONTROLLER,
              L_MESSAGE_MANAGER    TYPE REF TO IF_WD_MESSAGE_MANAGER.
        L_CURRENT_CONTROLLER ?= WD_THIS->WD_GET_API( ).
        L_MESSAGE_MANAGER = L_CURRENT_CONTROLLER->GET_MESSAGE_MANAGER( ).
    *     report message
        L_MESSAGE_MANAGER->REPORT_ATTRIBUTE_EXCEPTION(
            MESSAGE_OBJECT = L_EXC
            ELEMENT = LO_EL_NODE_COMPUTER
            ATTRIBUTE_NAME = WD_ASSIST->CONTEXT_ATTR
    *         MSG_USER_DATA    =
```

```
                    MESSAGE_TYPE     = L_MESSAGE_MANAGER->CO_TYPE_WARNING
                         ).
            ENDTRY.
         ENDMETHOD.
```

4）使用事务代码 SE61，编辑常规文本，如图 7-114 所示。

修改 一般测试：Z_NUMBER_HELP 语言 ZH

插入　行　格式化　页号

```
    ....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
*   数值判定分以下情况：
LZ
T1  0,  1       &<1       错误
T1  2...100     2<&<100   警告
T1  101...333   101<&<333 成功
T1  334...666   334<&<666 警告异常
T1  667...900   667<&<900 错误异常
T1  901...      901<&     严重错误
```

图 7-114

5）编辑 Web Dynpro Application 如图 7-115 所示。

Create Web Dynpro application

Application	zwd_message
Description	信息栏及信息应用

图 7-115

6）运行结果如下：

输入 1，如图 7-116 所示。

数字需大于1。

数字: 1 确认

图 7-116

输入 3，如图 7-117 所示。

数字在100以内。

数字: 3 确认

图 7-117

输入 113，如图 7-118 所示。

数字刚刚好!

数字: 113 确认

图 7-118

输入 335，如图 7-119 所示。

数字在334到666之间。

数字: 335 确认

图 7-119

输入 667，如图 7-120 所示。

数字在667到900之间。

数字: 667 确认

图 7-120

输入 1000，如图 7-121 所示。

数字 1000 大于900。

数字: 1,000 确认

图 7-121

7.7 Suspend 和 Resume 调用

在很多情况下，想定位到独立运行的第二个应用程序，还不能关闭第一个应用程序，在 Web Dynpro ABAP 程序中，挂起（Suspend）和恢复（Resume）调用是很有用的。要使用暂停和恢复调用，可以从一个 Web Dynpro 窗体的界面视图中选择任何出站或入站插头作为挂起插头（Suspend Plugs）或者恢复插头（Resume Plugs）。挂起插头调用类似于退出插头（Exit Plugs）调用，都是用于退出活动的 Web Dynpro ABAP 应用。区别于退出插头的是，它使用户在使用了另一个单独的应用程序之后返回到原来的应用程序。原来的应用只是暂时停用或暂停，不会被关闭。它可以通过调用恢复插头重新激活。

可以在 Web Dynpro ABAP 应用程序中设置一个挂起插头来调用以下程序：

1）另一个 Web Dynpro 应用程序。

2）使用不同的技术的 Web 应用程序（如 BSP）。

由于挂起和恢复插头用于活动的应用程序之间进行通信，它们必须添加到该窗体的接口视图中。每个挂起插头都赋有 URL 参数，参数在这里是强制性的，不是可有可无的，这一点不像退出插头。当一个挂起的插头被调用时，必须指定目标 URL。

如果至少有一个挂起插头在接口视图中创建，则也必须创建一个恢复插头。没有恢复插头，应用程序在挂起后不能被重新激活。每一个的 Web Dynpro 窗体中只可以有一个恢复插头可以被定义。

（1）挂起一个 Web Dynpro ABAP 应用程序

这是很容易设立一个挂起的调用：首先，在主 WDA 应用程序中选择出站插头作为挂起插头。调用插头时，作为 URL 参数的值指定一个字符串，表示目标应用程序的 URL（请参阅下面的示例代码），如图 7-122 所示。

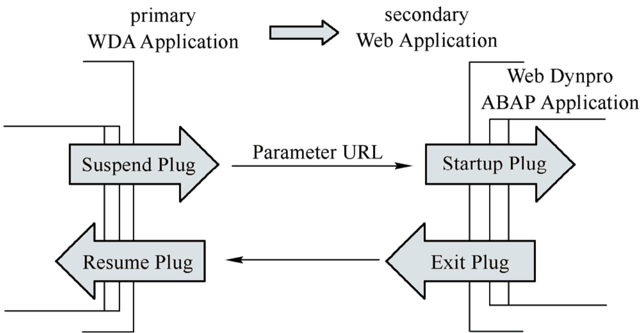


图 7-122

```
method ONACTIONGO_SUSPENDING .
data: L_REF_MAIN_WINDOW type ref to IG_MAIN_WINDOW .
L_REF_MAIN_WINDOW =     WD_THIS->GET_MAIN_WINDOW_CTR( ).
  L_REF_MAIN_WINDOW->FIRE_GO_SUSPENDING_PLG(
    URL =     'http://…/webdynpro/sap/my_secondary_wda'    ).

endmethod
```

Web Dynpro 应用程序的 URL 在相关的标签页中被定义为应用程序属性之一，它可以从这里访问。

在第二个 WDA 应用程序中，当退出插头触发时，该应用程序被关闭，并且自动调用主应用程序的恢复插头。在 Web Dynpro 框架内，主程序的活动实例的引用总是在挂起调用时转移。第二个应用程序知道它的状态，作为挂起的主应用程序随后的应用程序，在其中当使用指定引用的一个退出插头被调用时会返回到主应用程序的恢复插头。在这种情况下，没有进一步的编程的必要了。

（2）挂起一个非 Web Dynpro ABAP 应用程序

创建一个挂起的调用到一个非 WDA 的应用，因为也需要在第二个应用中修改，这是非常困难的。例如，程序员必须实施返回到主应用程序的机制，以及指定第二个应用程序的 URL

地址和强制性参数，还需要在随后返回步骤中指定自己应用程序的 URL。这就是所谓的恢复 URL，作为参数这也传递到第二个应用程序中，如图 7-123 所示。

URL 必须在第二个应用程序中被读取。从而在第二个应用程序的浏览器中重定向到恢复插头传过来的 URL，然后打开浏览器窗口中的二次应用。

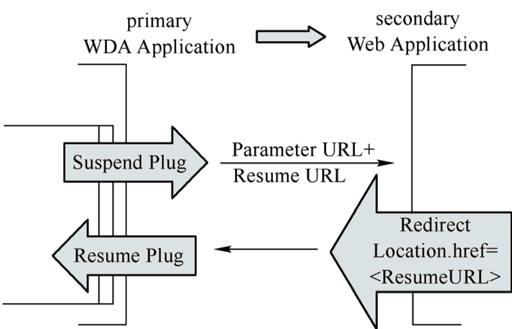


图 7-123

主程序通过恢复插头的 URL 被调用，被重新激活。

挂起和恢复插头，可以有用户定义的参数。

注：挂起和恢复调用在门户环境中没有发布。

下面介绍窗体中出、入站插头的参数。

（1）入站插头和出站插头类型为 Standard 的参数

对于每一个标准类型的出站插头，程序员可以创建一个用户定义的参数列表。为每个标准类型的入站插头，程序员可以在相关的事件处理程序方法中创建一个用户定义的输入参数列表。

（2）挂起插头和退出插头的参数

程序员可以为这些特殊的窗体插头定义参数。由于这些参数必须在输出插头的 URL 中指定，所以一定是 String 类型或 TIHTTPNVP 类型：

退出插头和挂起插头的所有参数在目标 URL 中指定（退出插头是可选的，挂起插头是强制的）。如果目标 URL 是 Web Dynpro 应用程序，则应符合下列格式：

```
<schema>://<host>.<domain>.<extension>:<port>/sap/bc/webdynpro/<Namespace>/<application name>?<parameter name>=<parameter value>
```

退出插头和挂起插头在运行时有特定的字符串参数 URL。

（3）启头插头和恢复插头的参数

启动插头和恢复插头的参数通过相关的事件处理程序方法调用的 URL 进行传递，这意味着参数值来自第二个应用。出于这个原因，传递的参数数目和名称在设计时对于主 Web Dynpro 应用程序来说通常是未知的。为了能将值传递到入站插头的 URL 参数中，它们存储在一个类型为 TIHTTPNVP 的表中。在该表中的所有应用程序特定的 URL 参数可以通过参数 WDEVENT 传递，该参数属于启动或恢复插头的事件处理程序方法。在本地创建的参数必须是相同的类型，即 TIHTTPNVP。

```
method HANDLEDEFAULT .
  data: l_url_parameters type tihttpnvp.
      .
      .
  wdevent->get_data(
    exporting
      name =    if_wd_application=>all_url_parameters
    importing
      value = l_url_parameters    ).
endmethod.
```

如果在设计时启动或恢复插头的参数是已知的，程序员可以将个别参数字符串添加到事件处理程序方法的参数中，并直接访问它们。

如果事件处理程序方法的入站插头有输入参数，则参数值必须在运行时是可用的。如果调用 URL 不指定参数值，也没有定义应用程序的任何默认值，则将导致一个运行时错误。（请参阅调用 Web Dynpro 应用参数）。通过定义默认值，在不指定参数值时可以防止运行时错误。

（4）TIHTTPNVP 参数

表类型 TIHTTPNVP 包含参数的名称和值。如果程序员创建一个这种类型的挂起插头或退出插头的参数，则该参数是一个表，其中包含字符串类型的名称和值。在这种情况下，所有项都被传递到 URL，而不是把每一个参数分开传递。

传递启动插头或退出插头的名称和值都存储在一个表中。

注：其 URL 参数的事件处理程序方法也提供了一个相关的参数，也以相同的表格类型存储在参数表中。处理应用程序对类型 TIHTTPNVP 表中传输参数进行检查和验证。

7.8 处理 Web 图标

当用户为 Web 应用程序创建界面时，具有固定大小并能多次使用的图形元素是需要的。例如使用按钮或屏幕列表中的图标或状态。这些图标作为一个对象或一个函数的图形图像。SAP 为使用最频繁的标准对象和函数提供了预定义的 Web 图标。

Web Dynpro ABAP 提供了各种技术来解决的 UI 元素中的 URL（如 Image.source）：

- 类 CL_WD_WEB_ICON 是用于访问 Web 图标的类。
- 新的 Web 图标可以通过~IconAlias 访问，其中包含支持的 URL 格式，见表 7-2。

表 7-2

格　式	说　明	翻　译	示　例
~IconAlias	新的预定义的 Web 图标的格式	大写字母	~Icon/Attachment ~IconLarge/AudioFile ~Pictogram/2PeopleDocCircularArrow
$NAME$ $NAME$/path	$NAME$是指事务 SM59 中的一个 HTTP 连接（G 型）条目。如果指定了路径，则它被附加到已定义路径的前缀。这是使用绝对 URL 的唯一方法	$NAME$ 为大写字母，/path 为小写字母	$MY_HTTP_DESTINATION$/images/hello.gif

（续）

格　　式	说　　明	翻　　译	示　　例
/path	从服务器到一个 MIME 信息库相对路径信息入口。这个入口以 MIME 库根目录开始。服务器的 URL 总是加载。如果路径是一个 Web Dynpro 的路径，则该路径转换为一个图标，否则没有方法确保该 URL 的有效性	小写字母	/sap/public/images/hello.gif
image.ext folder/image.ext	从一个活动组件的 MIME 文件夹加载图片，这是短格式		s_lr_11417.jpg hello.gif images/hello.gif
{/NAMESPACE/COMPONENT}/image.ext {/NAMESPACE/COMPONENT}/folder/image.ext	图像指定了另一个组件的相对路径，也可以是一个单独的组件	命名空间和组件为大写字母，路径为小写字母	{WDR_TEST_EVENTS}/s_lr_11417.jpg

- 新图标在早期的版本中的一个子集中，并以 WEBICON_*作为头字母，见表 7-3。

表 7-3

格　　式	说　　明	翻　　译	示　　例
ICON_<NAME>	SAP GUI 图标，不能用于 Web	大写字母	ICON_CRITICAL
WEBICON_<NAME>	该格式已被符号~IconAlias 取代	大写字母	WEBICON_CRITICAL
[WWWDATA]/objectName	从 SAP Web 的 ITS 库中加载图片		[WWWDATA]/HELLO_IMAGE

注：语法 WEBICON_*已过时。较早的 WEBICON_*名称内部被映射到新的 Web 图标。需要注意的是老的 SAP GUI 界面图标（ICON_*）不再被使用。

注：/路径，不支持相对路径，就是以下的格式./path 或 ../path。

下列公共 API 可用于程序员开发。

- 值帮助：功能模块 WDY_WB_VD_ICON_PICKER。
- 数据元素：网页图标别名 WDG_UR_WEB_ICON_ALIAS。
- 域名：网页图标别名 WDG_UR_WEB_ICON_ALIAS。

附录 A　Web Dynpro For ABAP 系统类

SAP 指定的用户接口控件开发了许多 Web 应用程序的 UI 元素。这确保用户将看到一个标准的屏幕，然而并非所有的 UI 元素显示在相同的格式中，这意味着某些动态编程是必需的。

一般规则是，以 CL_WD_或 IF_WD_作为头字母的类和接口已经为应用程序开发人员发布。

注：以 CL_WDR_或 IF_WDR_开始的类和接口尚未释放。这些类不包含在接口包中，它们可以更改，SAP 不另行通知。此外，使用这些内部类的方法调用有可能会产生副作用。

以下的编程接口支持应用程序开发人员进行动态编程：运行时 APIs、Context 节点 APIs、个性化 APIs、门户集成 APIs。

1．运行时 APIs

（1）IF_WD_ACTION

IF_WD_ACTION 接口使程序员能够访问控制器的 Action 的信息。

一个 UI 元素可以绑定到一个动作 Action。它是由一个特定的 UI 元素触发事件发生时执行的动作。如果事件发生时，则在相应的视图控制器实现的事件处理程序被调用。接口属性见表 A-1。

1）属性 Attributes：

表 A-1

属　性	类　型	描　述
CONTROLLER	实例属性	动作 Action 的控制器
IS_ENABLED	实例属性	指定当前 Action 是否为活动的。如果 Action 为活动的，则属性的值为‘X’，否则为‘ ’
IS_VALIDATING	实例属性	标示当前 Action 是否为校验或非校验的 Action。如果 Action 为校验的，则属性的值为‘X’，否则为‘ ’
IMAGE	实例属性	Action 的 Image 图标地址 URL 信息
PARAMETERS	实例属性	动作 Action 参数：表或值的名称
TEXT_KEY	实例属性	动作 Action 的标签文本

2）方法 SET。

此方法设置动作的属性。可设定以下可选参数。

- ENABLED：确定 Action 是否有效或无效。
- VALIDATING：确定 Action 是进行验证或不验证。
- IMAGE：确定 Action 图标 Image 的 URL 地址。
- TEXT：确定 Action 的文本标示。

（2）IF_WD_APPLICATION

IF_WD_APPLICATION 接口可以访问整个 Web Dynpro 应用中有效的数据。

方法：

1）GET_APPLICATION_INFO：返回应用程序的元数据描述的方法。结果是一个 IF_WD_RR_APPLICATION 类型的对象。

2）GET_IS_RTL：用于取得该应用程序的文本显示格式是否被切换到从右到左（RTL）的格式。

注：RTL 是来自中东的语言的文字方向。例如，希伯来文和阿拉伯文是从右到左。从右到左（RTL）的反面是从左到右（LTR），这是正常的印欧语系语顺。然而，在 RTL 语言中，数字（如电话号码或邮政编码）和适当的名称都是从左到右写。因此，在 RTL 语言中往往是 LTR 语言和部分 LTR 句子混合的 RTL 文本。

Web Dynpro 编程模型支持 RTL 概念。应用程序开发人员应注意以下几点去使用完整的 RTL 设置：

① 确保帧（Frame）出现在正确的顺序，即相比 LTR 帧的顺序，RTL 正好相反：

- 菜单和内容列表一般设为 RTL。
- 实际的内容一般设为 LTR。

② 如果需要，则可以创建图片 RTL 版本，并确保 RTL 版本的图像在应用程序中使用。

- GET_IS_ACCESSIBLE：用于取得数据可访问的模式是否被激活。
- OPEN_HELP_CENTER：用于打开帮助中心。
- GET_REMOTE_ADDRESS：取得当前调用程序的 IP 地址。
- GET_CLIENT_ENVIRONMENT：返回应用程序运行中客户端环境。例如，客户端环境的门户网站或 SAP NetWeaver 业务客户端。可以从 CO_CLIENT_ENVIRONMENT 常数值得到。
- GET_CONFIGURATION_ID：用于取得正在运行的应用程序的配置 ID。
- GET_IS_CLOSE_WINDOW_SUPPORTED：用于返回关闭一个窗口是否是可能的。

（3）IF_WD_COMPONENT

IF_WD_CONTROLLER 接口使程序员可以访问活动的 Web Dynpro 组件的信息。接口被类 CL_WDR_COMPONENT 实例化。该类是所有的 Web Dynpro 组件的基类。

在 Web Dynpro 组件中下面的方法调用返回一个类型为 IF_WD_COMPONENT 的组件控制器对象：

```
data:
    l_componentcontroller_api type ref to if_wd_component.
    l_componentcontroller_api = wd_this->wd_get_api( ).
```

方法：

1）GET_ID：用来返回一个唯一的组件 ID。

2）REMOVE_PENDING_INPUT：用来清空尚未被传送到 Context 中的用户输入。

3）GET_WINDOW_MANAGER：用来得到的窗体控制器。结果是一个类型为 IF_WD_WINDOW_MANAGER 的对象。

4）GET_COMPONENT_INFO：用来返回组件的元数据（Metadata）描述。结果是一个类型为 IF_WD_RR_COMPONENT 的对象。

5）GET_PORTAL_MANAGER：用来取得门户网站管理器的一个引用。结果是一个类型为 IF_WD_PORTAL_ INTEGRATION 的对象。

6）GET_CONFIGURATION_KEY：取得当前的配置密钥（或被设置为初始值）的方法。

7）ADD_EVENT_HANDLER：注册事件的事件处理程序的方法。

8）REMOVE_EVENT_HANDLER：注销事件的事件处理程序的方法。

9）CANCEL_NAVIGATION：禁止整个应用程序的导航的方法。

10）GET_PERSONALIZATION_MANAGER：该方法得到一个个性化管理器的引用。结果是一个类型为 IF_WD_PERSONALIZATION 的对象。

11）GET_APPLICATION：得到一个应用程序对象的引用。结果是一个类型为 IF_WD_APPLICATION 的对象。

12）GET_CMP_USAGE_GROUP：得到一个引用的组件的使用组中。结果是一个类型 IF_WD_COMPONENT_USAGE_ GROUP 对象。

13）CREATE_CMP_USAGE_GROUP：创建一个组件的使用组。结果是一个类型 IF_WD_COMPONENT_USAGE_GROUP 的对象。

14）HAS_CMP_USAGE_GROUP：确定组件是否存在使用组。

（4）IF_WD_COMPONENT_USAGE

IF_WD_COMPONENT_USAGE 接口使程序员访问方法用于操作（外部）的 Web Dynpro 组件的使用组件。

为了使用一个不同的 Web Dynpro 组件，必须维护组件本身的属性，以及想要访问的组件的属性。下面的方法调用后在该控制器中返回一个类型为 IF_WD_COMPONENT_USAGE 的对象。

```
data:
    component_usage type ref to if_wd_component_usage.
    component_usage = wd_this->wd_cpuse_<Name der Component-Usage>( ).
```

方法：

1）CREATE_COMPONENT：该方法在组件使用的点上用于创建 Web Dynpro 组件。如果一个激活的组件实例已经存在用于该组件的使用，调用该方法时会发生运行时错误。出于这个原因，在调用该方法之前使用方法 HAS_ACTIVE_COMPONENT，以检查该组件的一个活动实例是否已经存在。

2）DELETE_COMPONENT：该方法在组件使用的点上用于删除 Web Dynpro 组件。如果没有活动的组件实例存在的情况下，该方法不能被调用。

3）HAS_ACTIVE_COMPONENT：该方法判断指定的 Web Dynpro 组件是否在该组件使用点已经被创建，并且是否是活动的。

4）GET_IMPLEMENTING_IF_VIEW_INFO：该方法返回元数据描述的接口视图。结果返回一个类型为 IF_WD_RR_INTERFACE_ VIEW 的对象。

5）GET_INTERFACE_CONTROLLER：该方法返回一个 Web Dynpro 组件的接口控制器

的引用。结果返回类型取决于 Web Dynpro 组件的接口控制器中具体的定义，该类型可以在此控制器的属性中找到。

6）ADD_EVENT_HANDLER：该方法可以用于在运行时注册动态事件的事件处理程序。

7）CREATE_COMP_USAGE_OF_SAME_TYPE：该方法创建一个指向同一个组件的组件使用，结果是一个类型为 IF_WD_COMPONENT_USAGE 的对象。

8）ENTER_REFERENCING_MODE：可以使用该方法来确定一个组件使用指向一个不同的组件。

注：

① 组件的使用应该指向不同的组件（引用组件的使用），并且该组件不能有任何活动的组件使用。

② 必须参照相同的组件作为引用组件的使用。

9）GET_COMPONENT_USAGE_INFO：该方法返回元数据描述的组件使用。结果是一个类型为 IF_WD_RR_COMPONENT_ USAGE 的对象。

10）IS_REFERENCING：该方法指定组件使用是否指向一个不同的组件使用。其结果可能是 X（组件的使用指向不同的组件）或“”（组件的使用不涉及不同的组件使用）。

11）IS_REFERENCED：该方法指定组件使用是否由不同的组件使用作为参照。其结果可以是 X（组件的使用通过一个不同的组件使用被引用）或“”（组件使用未引用不同的组件使用）。

12）LEAVE_REFERENCING_MODE：使该组件的使用不再是指向另一个不同的组件使用。

13）REMOVE_EVENT_HANDLER：可以用于在运行时动态注销一个事件处理程序的事件。

（5）IF_WD_COMPONENT_USAGE_GROUP

一个组件使用组将组件的使用组织到一起，使用相同的组件及组件接口的用法。

利用组件使用组使得它更容易动态地创建和管理组件。

方法：

1）GET_COMPONENT_USAGE：该方法为组内的一个组件使用的检索方法。结果是一个类型为 IF_WD_COMPONENT_ USAGE 的对象。

2）GET_COMPONENT_USAGES：该方法用于返回组内的所有组件使用。

3）ADD_COMPONENT_USAGE：该方法创建一个组件的使用并添加组件的使用到组内。

4）REMOVE_COMPONENT_USAGE：该方法从组中删除组件的使用。

5）REMOVE_ALL_CMP_USAGES：该方法从组中删除所有组件使用。

6）SET：该方法设置组件使用的属性。

（6）IF_WD_CONTROLLER

IF_WD_CONTROLLER 接口，利用它可以通过 Web Dynpro 控制器能够访问相关信息。

下面的方法调用返回一个对象，该对象的类型为 Web Dynpro 组件的组件控制器内的 IF_WD_CONTROLLER：

```
data:
  l_api     type ref to if_wd_controller.
  l_api = wd_this->wd_get_api( ).
```

方法：

1）GET_MESSAGE_MANAGER：该方法返回一个消息管理器的引用。结果是一个类型为 IF_WD_MESSAGE_MANAGER 的对象。

2）GET_COMPONENT：该方法返回一个属于该控制器的组件引用。结果是一个类型为 IF_WD_COMPONENT 的对象。

3）GET_CONTEXT：该方法返回一个属于该控制器的 Context 的引用。结果是一个类型为 IF_WD_CONTEXT 的对象。

4）GET_MESSAGE：该方法返回消息文本给消息关键字。

5）GET_ACTION：该方法返回一个该控制器的动作 Action 的引用。结果是一个类型为 IF_WD_ACTION 的对象。

6）GET_CONTROLLER_INFO：该方法返回一个该控制器的元数据的引用。结果是一个类型为 IF_WD_RR_CONTROLLER 的对象。

7）IF_WD_MESSAGE_MANAGER：该接口作为一个信息集成的消息管理器，为应用程序开发者提供 Web Dynpro 代码向导。

下面列出比较重要的方法。如果需要了解这些方法中的各个参数的描述，请参见系统中的相关方法的文档。

方法：

需要注意的是，RAISE 方法已经过时。出于这个原因，程序员不应该使用它们。另外，大多数方法使用的参数 msg_user_data 仅可用于特殊目的。出于这个原因，程序员不应该使用它。相关方法，可以使用可选 VIEW 参数输入要在其中显示的信息的窗体或视图的名称。对于 REPORT_方法 ENABLE_MESSAGE_NAVIGATION 参数信息，可参见 MessageArea 事件（MessageArea Events）。

因此，应用程序的消息在请求和返回期间不保存在消息管理器中，一个消息处理的机制（这些消息必须在每次往返过程中建立新的）内，非一时性消息是可用的。这使得一个消息的有效期可能为组件的生命周期、控制器的生命周期、Context 的生命周期（只适用于相关消息 Context 的属性）。

IS_PERMANENT 和 SCOPE_PERMANENT_MSG 参数的应用如下。

1）REPORT_EXCEPTION：报告一个 Web Dynpro 异常（可以回到原画面）。

2）REPORT_SUCCESS：报告成功的消息（■）。

REPORT_WARNING：报告警告的消息（⚠）。

REPORT_ATTRIBUTE_ERROR_MESSAGE：报告一个 Context 属性相关的 Web Dynpro 异常。该方法为 Context 属性报告一条错误消息。一个 Context 相关的错误消息影响的 Web Dynpro 阶段模型的流量。如果这种类型的错误信息在操作处理程序之前被调用（如在 WDDOBEFOREACTION 方法），则标准（standard）类型的动作不再被执行。随后的导航和 WDDOMODIFYVIEW 方法也不会被执行。然而，独立验证操作被执行时，即使存在错误的消息，后续的导航及 WDDOMODIFYVIEW 方法还是执行。

REPORT_ATTRIBUTE_EXCEPTION：报告一个 Context 属性的异常。该方法报告一条 Context 属性的 Web Dynpro 错误消息，同方法 REPORT_ATTRIBUTE_ERROR_MESSAGE。

REPORT_ERROR_MESSAGE：报告一个带可选参数的 Web Dynpro 的错误消息（ ）。在这里，可以传输一个字符串。该字符串可以是一个用户定义的 OTR 文本或一个异常类的文本。使用硬编码，以确保它可以很容易地转换为文本。

IS_EMPTY：查询是否有任何消息。

REPORT_T100_MESSAGE：报告一个消息，使用 T100 条目。

REPORT_FATAL_EXCEPTION：该方法报告一个终止消息。这种类型的消息在下面的方法中抛出会导致 Web Dynpro 应用程序因运行时错误而终止：

- WDDOINIT。
- WDDOEXIT。
- WDDOMODIFYVIEW。
- 在一个入站事件处理程序中。

如果这种类型的消息生成在 WDDOBEFORENAVIGATION 或 WDDOPOSTPROCESSING 方法或在一个动作处理程序中，则处理在该点被终止，并显示一个错误消息。

REPORT_FATAL_ERROR_MESSAGE：报告一个带有可选参数的致命的 WDA 消息。功能和现象同 REPORT_FATAL_ EXCEPTION。

CLEAR_MESSAGES：删除所有信息。

REPORT_ATTRIBUTE_T100_MESSAGE：报告一个 Context 属性的 Web Dynpro 异常。

REPORT_ELEMENT_ERROR_MESSAGE：报告一个 Context 属性的 Web Dynpro 例外。

这种方法报告一条错误消息的 Context 元素。一个 Context 相关的错误消息影响的 Web Dynpro 阶段模型的流向。如果这种错误信息类型（E 型）的操作处理程序在显示之前被调用（如在 WDDOBEFOREACTION 方法中），则标准类型（standard）的行动（Action）将不再被执行，随后的导航和 WDDOMODIFYVIEW 方法也不会被执行。然而，独立验证操作会被执行，即使存在错误的消息，后续的导航及 WDDOMODIFYVIEW 方法还是执行。

3）REPORT_ELEMENT_EXCEPTION：报告一个 Context 属性的 Web Dynpro 例外。功能和现象同 REPORT_ELEMENT_ ERROR_MESSAGE。

4）REPORT_ELEMENT_T100_MESSAGE：报告一个 Context 属性的 Web Dynpro 例外。功能和现象同 REPORT_ELEMENT_ERROR_ MESSAGE。

5）REMOVE_MESSAGE：该方法将删除指定 ID 的消息。

6）HAS_VAL_ERRORS_FOR_WINDOW：可以使用该方法，以确定一个特定的窗体中是否有 Context 引用的错误消息。

7）HAS_VALIDATION_ERRORS：可以使用该方法，以检查在一个 Context 引用中是否有错误消息。

8）GET_MESSAGES：该方法返回一个所有可用消息的列表。

9）GET_MESSAGE_FOR_ID：该方法返回相关消息的信息。

（8）IF_WD_NAVIGATION_SERVICES

该接口为动态导航提供服务。

方法：

1）PREPARE_DYNAMIC_NAVIGATION：此方法创建一个指定参数的导航链接和导航目标，同时建立必要的视图嵌入和视图容器分配。如果目标视图来自一个嵌入式组件，如果需要，则创建该组件使用。这种方法被调用后导航可以执行，激活输出插件中指定的SOURCE_PLUG_NAME。

下面是传入参数：

- SOURCE_WINDOW_NAME：用于导航的窗体名称，在这个窗体中嵌入开始和目标视图。
- SOURCE_VUSAGE_NAME：嵌入的开始视图名称。
- SOURCE_PLUG_NAME：出站插头的名称。
- TARGET_COMPONENT_NAME：目标组件的名称，即在内部的窗体中显示的视图。
- TARGET_VIEW_NAME：进行导航的目标视图名称。
- TARGET_PLUG_NAME：入站插头的名称
- TARGET_EMBEDDING_POSITION：以下列格式嵌入视图的位置。

［<View1>/<Container1>.[<View2>/<Container2>] *].

如果该参数被设置为初始，则进行导航的视图将被直接嵌入在窗体中。

返回参数如下：

- COMPONENT_USAGE

组件使用的引用，如果视图来自外部组件则返回组件使用的引用。

2）DO_DYNAMIC_ NAVIGATION：该方法创建一个导航链接，并进行导航。

要做到这一点，需为导航创建导航链接、导航目标、出站插头、组件使用、视图嵌入和视图容器组件（这些还不存在），然后进行导航。

下面是传入参数：

- SOURCE_WINDOW_NAME：用于导航的窗体名称，在一个窗体中嵌入开始和目标视图。
- SOURCE_VUSAGE_NAME：嵌入的开始视图的名称。
- SOURCE_PLUG_NAME：出站插头的名称。
- PLUG_PARAMETERS：自定义事件的参数列表。
- TARGET_COMPONENT_NAME：目标组件的名称，即在内部的窗体中显示的视图。
- TARGET_VIEW_NAME：进行导航的目标视图名称。
- TARGET_PLUG_NAME：入站插头的名称。
- TARGET_EMBEDDING_POSITION：以下列格式嵌入视图的位置。

[<View1>/ <Container1>.[<View2>/<Container2>]*].

如果该参数被设置为初始，则进行导航的视图将被直接嵌入在窗体中。

下面是返回参数：

COMPONENT_USAGE：引用组件使用情况，如果视图来自外部组件，则返回组件使用的引用。

3）REMOVE_DYNAMIC_META_DATA：该方法将删除所有动态创建的元数据（查看嵌入、分配视图容器、导航链接、导航目标、组件的用法和出站插头）嵌入路径。

以下的进口中存在的参数：

- SOURCE_WINDOW_NAME：用于导航的窗体名称，在一个窗体中嵌入开始和目标视图。

- SOURCE_VUSAGE_NAME：嵌入的初始视图的名称。
- SOURCE_PLUG_NAME：出站插头的名称。
- TARGET_COMPONENT_NAME：目标组件的名称，即在内部的窗体中显示的视图。
- TARGET_VIEW_NAME：进行导航的目标视图名称。
- TARGET_PLUG_NAME：入站插头的名称。
- TARGET_EMBEDDING_POSITION：以下列格式嵌入视图的位置。

[<View1>/<Container1>. [<View2>/<Container2>] *].

（9）IF_WD_NAVIGATION_SERVICES_NEW

IF_WD_NAVIGATION_SERVICES_NEW 接口使程序员可以创建所需的元数据进行动态导航。这使得在设计时没有必要对导航链接进行声明。不过，这应该只用在一个静态声明的导航链接不可能实现时。在有些情况下，只有在运行时才能确定视图或嵌入式组件要显示。

接口 IF_WD_NAVIGATION_SERVICES_NEW 可被 Web Dynpro 组件中的一个视图调用，如下所示：

```
data:
l_view_api                    type ref to
  if_wd_view_controller,
l_navigation_services_new type ref to
  if_wd_navigation_services_new.
l_view_api = wd_this->wd_get_api(
  ).
l_navigation_services_new ?= l_api.
```

方法：

1）PREPARE_DYNAMIC_NAVIGATION：该方法创建一个指定参数的导航链接和导航目标，同时建立必要的视图嵌入和视图容器分配。如果目标视图来自一个嵌入式组件，如果需要，则创建该组件使用。这种方法被调用后导航可以执行，激活输出插件中指定的 SOURCE_PLUG_NAME。

调用该方法也导致对象类型 IF_WD_REPOSITORY_HANDLE 被创建，返回给调用者。该对象允许访问元数据时动态创建的方法被调用。

下面是输入参数：

- SOURCE_WINDOW_NAME：用于导航的窗体名称，在一个窗体中嵌入开始和目标视图。
- SOURCE_VUSAGE_NAME：嵌入的初始视图的名称。
- SOURCE_PLUG_NAME：出站插头的名称。
- TARGET_COMPONENT_NAME：目标组件的名称，即在内部的窗体中显示的视图。
- TARGET_VIEW_NAME：进行导航的目标视图名称。
- TARGET_PLUG_NAME：入站插头的名称。
- TARGET_EMBEDDING_POSITION：以下列格式嵌入视图的位置。

[<View1>/<Container1>. [<View2>/<Container2>] *].

如果该参数被设置为初始，进行导航的视图将被直接嵌入在窗体中。

下面是返回参数：

REPOSITORY_HANDLE (类型 IF_WD_REPOSITORY_HANDLE)：在其中所产生的元数据可以被访问的处理。

2）DO_DYNAMIC_NAVIGATION：

动态进行导航。要做到这一点，需为导航创建导航链接、导航目标、出站插头、组件使用、视图嵌入和视图容器组件（这些还不存在），然后进行导航。

调用该方法也导致对象的类型 IF_WD_REPOSITORY_HANDLE 被创建，返回给调用者。该对象允许访问元数据时动态创建的方法被调用。

注：使用相同的出站插头（SOURCE_PLUG_NAME）重复调用该方法。由于在该出站插头创建了多个导航链接，因此所有这些导航链接在随后的导航阶段中，可能导致显示的不是预期的视图，而是其他的视图。

以下的进口中存在的参数：

- SOURCE_WINDOW_NAME：用于导航的窗体名称，在这是一个窗体中嵌入开始和目标视图。
- SOURCE_VUSAGE_NAME：嵌入的初始视图的名称。
- SOURCE_PLUG_NAME：出站插头的名称。
- PLUG_PARAMETERS：出站插头的列表中的参数值。
- TARGET_COMPONENT_NAME：目标组件的名称，即在内部的窗体中显示的视图。
- TARGET_VIEW_NAME：进行导航的目标视图名称。
- TARGET_PLUG_NAME：入站插头的名称。
- TARGET_EMBEDDING_POSITION：以下列格式嵌入视图的位置。

[<View1>/<Container1>.[<View2>/<Container2>] *].

如果该参数被设置为初始，进行导航的视图将被直接嵌入在窗体中。

下面是返回参数：

REPOSITORY_HANDLE (类型 IF_WD_REPOSITORY_HANDLE)：在其中所产生的元数据可以被访问的处理。

（10）运行时库 APIs

运行时库由一系列接口组成，这些接口可以读取一个 Web Dynpro 应用程序的元数据。API 还允许动态元数据的创建，如在运行时创建组件使用或导航链接。

Web Dynpro 运行时库由以下接口组成：IF_WD_REPOSITORY_HANDLE、IF_WD_RR_APPLICATION、IF_WD_RR_CMP_USAGE_GROUP、IF_WD_RR_COMPONENT、IF_WD_RR_COMPONENT_USAGE、IF_WD_RR_CONTROLLER、IF_WD_RR_VIEW、IF_WD_RR_VIEW_CNT_ASSIGNMENT、IF_WD_RR_VIEW_USAGE、IF_WD_RR_WINDOW。

（11）IF_WD_VALIDATION

接口 IF_WD_VALIDATION 提供访问 Context 节点属性的各种 Web Dynpro 验证服务。

- 读取验证信息。
- 设置验证信息。

方法：

1）IS_ATTRIBUTE_VALID：检查属性是否是有效的。

2）IS_NODE_VALID：检查某节点的所有元素是否是有效的。

3）IS_ELEMENT_VALID：检查某元素所有属性是否是有效的。

4）IS_CONTEXT_VALID：检查根节点 Context 的所有节点是否是有效的。

5）SET_ATTRIBUTE_VALID：设置属性为有效。

6）SET_NODE_VALID：设置节点为有效。

7）SET_ELEMENT_VALID：设置节点元素为有效。

8）SET_CONTEXT_VALID：设置 Context 根节点为有效。

9）GET_MESSAGES_FOR_CONTEXT：提取 Context 所有的消息字符串。

10）GET_MESSAGES_FOR_NODE：提取某 Context 节点所有的消息字符串。

11）GET_MESSAGES_FOR_ELEMENT：提取某 Context 元素所有的消息字符串。

12）GET_MESSAGES_FOR_ATTRIBUTE：提取某 Context 属性所有的消息字符串。

（12）IF_WD_VIEW

接口 IF_WD_VIEW 是所有的 Web Dynpro 视图的通用接口。

方法：

1）GET_ELEMENT：该方法返回一个视图 UI 元素。该元素被指定一个 ID，或者返回 NULL，如果视图 UI 元素不存在。

2）GET_ROOT_ELEMENT：该方法返回相应的视图的元素树的根元素。

3）GET_VIEW_USAGE：此方法返回嵌入视图的元数据的描述。

4）REQUEST_FOCUS_ON_VIEW_ELEM：该方法将焦点设置在一个特定的视图元素上面。

5）RESET_VIEW：该方法重置视图布局，即将单独的视图的视图元素恢复到在设计时声明的初始状态。当它被初始化时所有视图的动态布局的修改被恢复。

6）SCROLL_INTO_VIEW：如果参数 WDLIGHTSPEED 被激活时，该方法放置一个视图元素到一个浏览器的可视区域。

注：该方法只能在方法 WDDOMODIFYVIEW 内被调用。

（13）IF_WD_VIEW_CONTROLLER

IF_WD_VIEW_CONTROLLER 接口使程序员可以使用 Web Dynpro 视图控制器访问相关信息。一般由类 CL_WDR_VIEW 实施，是接口 IF_WD_CONTROLLER 的扩展。

方法：

1）FIRE_PLUG：该方法一般用于触发动态生成的出站插头。

2）GET_VIEW_INFO：该方法返回视图元数据对象的一个引用。

3）REQUEST_FOCUS：该方法用于 Context 中的元素设置焦点。

4）REQUEST_FOCUS_ON_ACTION：该方法用于设置焦点到 UI 元素，以便触发特定的动作。

5）GET_CURRENT_ACTION：该方法返回正在处理的动作。

6）GET_EMBEDDING_WINDOW_CTLR：该方法返回已嵌入的窗口控制器的引用。结果是一个类型为 IF_WD_WINDOW_ CONTROLLER 的对象。

（14）IF_WD_VIEW_ELEMENT

接口 IF_WD_VIEW_ELEMENT 是在视图中的任何元素的接口。该接口用来唯一标识一个元素，并提供相应的视图。

当创建一个元素时，分配一个 ID 标识到 UI 元素，用于标示视图内的元素。

方法：

1）GET_ID：返回元素的唯一名称（ID）。

2）GET_VIEW：返回 UI 元素所在的视图。

（15）IF_WD_WINDOW

接口 IF_WD_WINDOW 用于一个弹出窗体的打开、关闭及编辑。

方法：

1）CLOSE：该方法异步关闭弹出“Phase Model Cycle”下完成的窗体。

2）OPEN：该方法以异步方式打开弹出窗体。

3）SET_WINDOW_SIZE：设置窗体的大小。

4）SET_WINDOW_TITLE：设置窗体的标题。

5）SET_ON_CLOSE_ACTION：该方法用于当单击“取消”时注册用户动作，。

6）SUBSCRIBE_TO_BUTTON_EVENT：用于当单击按钮时注册用户动作。

7）UNSUBSCRIBE_FROM_BUTTON_EVENT：用于注销用户的动作。

8）SET_REMOVE_ON_CLOSE：该方法用于关闭窗体后删除窗体。

9）SET_BUTTON_KIND：该方法用于指定按钮的组合。

10）SET_MESSAGE_TYPE：该方法用于设置消息类型。

11）SET_CLOSE_IN_ANY_CASE：该方法用于当单击任何窗体按钮时自动关闭窗体。

12）SET_BUTTON_ENABLED：该方法使按钮被激活和停用。

语法：

```
data: l_window type ref to if_wd_window.
l_window->set_button_enabled(
        button = <button>
       is_enabled = abap_false ).
```

源码中<button>代表一个弹出按钮，可以用常量 if_wd_window=>co_button 定义。

（16）IF_WD_WINDOW_CONTROLLER

使用接口 IF_WD_WINDOW_CONTROLLER 访问一个窗体的属性。

方法：

1）GET_WINDOW：该方法用于返回弹出窗口，如果存在。结果是一个类型为 IF_WD_WINDOW 的对象。

2）GET_WINDOW_INFO：该方法用来返回窗体的元数据描述。

（17）IF_WD_WINDOW_MANAGER

使用接口 IF_WD_WINDOW_MANAGER 可以创建不同的窗体和弹出对话框。

方法：

1）CREATE_EXTERNAL_WINDOW：创建一个外部浏览器窗体的方法。但是，如果应

用程序运行在门户网站中，则在外部浏览器中创建窗体是不可能的。该方法用来创建一个模态的弹出窗体。

注：模态参数已经过时了，因为 Web Dynpro 中没有外部模态窗体，目前不支持标题参数。

2）CREATE_WINDOW_FOR_CMP_USAGE：该方法用来使用组件的接口视图显示一个弹出窗体。

3）CREATE_POPUP_TO_CONFIRM：该方法用于创建一个弹出确认对话框。

（18）CL_WD_CUSTOM_EVENT

该类可以访问到动作或 UI 元素的 Web Dynpro 事件的参数。动作或事件的事件处理程序有一个该类型的输入参数。这个类的方法允许访问事件或动作的参数。

2．Context 节点 APIs

下面的接口和 ABAP 字典结构用于 Web Dynpro 中 Context 节点数据的存取：IF_WD_CONTEXT、IF_WD_CONTEXT_NODE、IF_WD_CONTEXT_NODE_INFO、IF_WD_CONTEXT_ELEMENT、结构 WDR_CONTEXT_ATTRIBUTE_INFO。

如果有一个错误，Context 处理方法总是会触发 CX_WD_CONTEXT，但该异常并不声明。

（1）IF_WD_CONTEXT

接口 IF_WD_CONTEXT 允许访问的一个控制器的 Context 节点的数据，并提供了用于记录用户输入的方法。

方法：

1）RESET_CHANGED_BY_CLIENT：该方法已过时，不应该使用它。

2）CONTEXT_CHANGE_LOG。

（2）IF_WD_CONTEXT_NODE

接口 IF_WD_CONTEXT_NODE 使一个单一的 Context 节点绑定数据或从 Context 节点中读取数据。

方法：

1）BIND_ELEMENT：该方法绑定 Context 元素到 Context 节点上，具体参数见表 A-2。

表 A-2

参数名称	类型	描述
NEW_ITEM	输入参数	该参数可以是一个 Context 元素的引用（类型为 IF_WD_CONTEXT_ELEMENT）或数据结构的引用，或数据结构本身。如果它是一个数据结构，考虑到性能方面的原因，应该使用 BIND_STRUCTURE
SET_INITIAL_ELEMENTS	输入参数	如果该参数标准值= ABAP_TRUE，则该节点的内容被释放，即该节点无效，然后有新元素插入。否则，该元素被添加到列表中
INDEX	输入参数	在指定元素之前添加新元素（SET_INITIAL_ELEMENTS = ABAP_FALSE 时）。该值为标准值 0 时添加新元素到最后一个元素
ELEMENT	返回参数	返回新的元素引用

调用方法时如果有错误，一个异常类型 CX_WD_CONTEXT 被触发。

2）BIND_ELEMENTS：该方法绑定 Context 元素到 Context 节点上。该方法本质上与方

法 BIND_ELEMENT 是相同的，区别是必须指定一个内部表，其中可以包含任一元素引用、数据引用或数据结构。

3）BIND_STRUCTURE：使用该方法，可以根据一个结构在节点中创建一个新的元素。该结构应该与优选作为节点声明的结构是相同的，但它不是必须的。如果该结构是不一样的，则对应的移动（move-corresponding）被执行。在其他方面，该方法的行为与 BIND_ELEMENT 一样。

4）BIND_TABLE：使用该方法绑定内部表到 Context 节点。该方法是 BIND_ELEMENTS 的优化版本。该表包含预期的数据。如果该表的数据结构与节点声明的结构相匹配，则下面的内部优化将被执行：只有表本身传递到节点。没有任何元素实例被创建。如果访问一个或多个元素，则这些元素的引用被创建。

在 ABAP 原则“Copy-on-write”用于传输复制的表时，这意味着只有当它被改变时该表被复制，而不是在它被传输时被复制。

注：只有索引表，即标准表和排序表支持该项操作。

5）CLEAR_SELECTION：当前的选择被清除。

6）CREATE_ELEMENT：使用该方法创建元素的引用，以后可以使用 BIND_ELEMENT 方法将其追加到节点。该元素的数据可以在追加到节点时附加，也可以在以后设置。创建成功后该元素的引用被返回。

7）CREATE_ELEMENT_STATIC_ATTR：该方法已过时，因此不使用它。

8）GET_ATTRIBUTE：该方法返回一个 Context 属性的值。

9）GET_ATTRIBUTE_REF：该方法返回一个静态或动态属性的引用及属性值。一个新的引用被创建。这意味着所做的内容变更的引用不改变属性本身。

10）GET_CHILD_NODE：该方法返回一个特定的子节点。

11）GET_CHILD_NODES：该方法返回所有子节点。

12）GET_CONTEXT：该方法返回引用控制器 Context（Context 节点）。特别是有关环境 Context 变化日志的引用。

13）GET_ELEMENT：该方法返回引用了一个元素，该元素要么是一个特殊的索引的元素，要么是头选择的元素。

14）GET_ELEMENTS：该方法返回一个所有元素引用的内部表。

15）GET_ELEMENT_COUNT：该方法返回元素的数量。

16）GET_LEAD_SELECTION：该方法返回头选择元素的引用。

17）GET_LEAD_SELECTION_INDEX：该方法返回头选择元素的索引。

18）GET_META_PATH：该方法返回在元数据中节点的路径。该路径不包含任何索引。

19）GET_NODE_INFO：该方法返回元数据对象的引用（IF_WD_CONTEXT_NODE_INFO）。

20）GET_PARENT_ELEMENT：该方法返回某节点的父节点所属的元素的引用。

21）GET_PATH：该方法返回原控制器中节点的路径。该路径包含节点名称和索引。

22）GET_SELECTED_ELEMENTS：该方法返回所有选中的元素的列表。

23）GET_STATIC_ATTRIBUTES：该方法返回所有声明为静态属性的副本。

24）GET_STATIC_ATTRIBUTES_REF：该方法返回所有静态属性的副本的引用。

25）GET_STATIC_ATTRIBUTES_TABLE：该方法返回所有的静态属性的所有元素。

26）INVALIDATE：该方法无初始化节点的所有元素。这意味着，所有的元素和子节点的实例被删除。初始化后节点的状态是它的初始状态。当一个元素或一个属性被再次访问，如果供给方法是可用的，那么供给方法会被调用。

27）IS_ALIVE：该方法检查相关的控制器是否还在生命周期内，这意味着它仍然可以被访问，并没有被 WDDOEXIT 方法终止。

IS_ATTRIBUTE_NULL
IS_CHANGED_BY_CLIENT
IS_FINALIZED

28）IS_SELECTED：该方法用于判断带有索引相关联的元素是否被选择。

29）IS_SUPPLIED：该方法用于判断节点是否填充或仍处于初始状态（或失效）。

30）MOVE_ELEMENT：该方法设置头选择到某一个元素。

31）MOVE_FIRST：该方法设置头选择到第一个元素。

32）MOVE_LAST：该方法设置头选择到最后一个元素。

33）MOVE_NEXT：该方法设置了头选择到下一个元素。

34）MOVE_PREVIOUS：该方法设置头选择为上一个元素。

35）MOVE_TO：该方法设置头选择为指定索引的元素。

36）PATH_GET_ATTRIBUTE：该方法返回数据的属性的引用。该引用可从 ME 节点出发通过相对路径访问。如果有一个错误，则会触发异常类 CX_WD_CONTEXT。

37）PATH_GET_ELEMENT：该方法本质上与 PATH_GET_ATTRIBUTE 相同，但不返回一个元素引用。

38）PATH_GET_NODE：该方法本质上与 PATH_GET_ATTRIBUTE 相同，但不返回一个节点的引用。

39）REMOVE_ELEMENT：该方法从节点中删除一个元素。该元素可以在以后再次追加到相同的节点。

RESET_CHANGED_BY_CLIENT
SET_ATTRIBUTE
SET_ATTRIBUTE_NULL
SET_CHANGED_BY_CLIENT
SET_FINALIZED
SET_LEAD_SELECTION
SET_LEAD_SELECTION_INDEX
SET_SELECTED
SET_STATIC_ATTRIBUTES
SET_STATIC_ATTRIBUTES_NULL

40）TO_XML：该方法对节点进行 XML 序列化。

（3）IF_WD_CONTEXT_NODE_INFO

IF_WD_CONTEXT_NODE_INFO 是 Context 节点的元数据的接口，被用来描述数据。

每个 Context 节点包含一个有元数据节点信息的引用。下面是最重要的元数据，可以使用 IF_WD_CONTEXT_NODE_INFO 访问：

- 名称（GET_NAME）。
- 单节点（IS_SINGLETON）。
- 多节点（IS_MULTIPLE）。
- 强制性的节点（IS_MANDATORY）。
- 强制性的选择（IS_MANDATORY_SELECTION）。
- 多选（IS_MULTIPLE_SELECTION）。

1）GET 方法：获取 Context 节点和子节点的属性。使用 GET 方法来返回单个组件的 Context 节点，见表 A-3。

表 A-3

方　　法	描　　述
GET_ATTRIBUTE GET_ATTRIBUTES	获取一个或全部属性的信息 使用 GET_ATTRIBUTE 方法可以返回 WDR_CONTEXT_ATTRIBUTE_INFO 属性的引用。使用 GET_ATTRIBUTES 返回一个类型为 WDR_CONTEXT_ATTR_INFO_MAP 的对象，这是一个类型为 WDR_CONTEXT_ATTR_INFO 表
GET_ATTRIBUTE_NAMES	该方法以 STRING_TABLE 类型的对象返回所有属性的名称 这种方法要比 GET_ATTRIBUTES 方法使用较少的时间和较少的内存，因为它没有提供太多的信息。如果只需要属性的名称，则使用 GET_ATTRIBUTE_ NAMES 方法
GET_ATTRIBUTE_FORMAT_PROPS	该方法读取特殊格式属性的特性
GET_CHILD_NODE GET_CHILD_NODES	返回一个或所有子节点 GET_CHILD_NODE 取得一个子节点的引用，GET_CHILD_NODES 取得一个列表，其中为所有子节点的名称和引用
GET_ATTRIBUTE_VALUE_HELP	该方法得到相应名称的属性的输入帮助
GET_ATTRIBUTE_VALUEHELP_TYPE	返回输入帮助的类型 类型可以是一个 ABAP 字典搜索帮助、OVS 输入帮助，甚至一个用户定义的输入帮助。该输入帮助也可以是取消激活的
GET_CONTROLLER	返回相关的控制器的方法，其中节点已被声明
GET_NAME	返回 Context 节点的名称
GET_PARENT	返回 Context 节点的父节点。该方法返回父节点的引用
GET_PRIMARY_ATTRIBUTE	基本属性是唯一重要的配置，因为在运行过程中不适用，只在配置框架时才支持 如果节点存在，则该方法返回该节点的基本属性的名称
GET_STATIC_ATTRIBUTES_TYPE	该方法从 RTTI 实例中检索静态声明的属性的节点。返回节点的 RTTI_ Reference 类型。了解 RTTI 的更多信息，请参阅类 CL_ABAP_TYPEDESCR 的相关文档
FOLLOW_PATH	该方法用于按照完整的路径表达式，程序员可以访问相关的目标节点。使用这个方法，可以在层次结构中从另一个节点之下的第一个节点（子节点）得到相关节点的引用

注：单个组件返回单一的名称，多个组件则返回一个列表。

2）HAS/IS 方法：被用来返回 Context 节点的不同的属性和状态，见表 A-4。

表 A-4

方　　法	描　　述
HAS_ATTRIBUTE_VALUE_HELP	查询 Context 属性输入帮助查询是否可用
IS_ALIVE	检查相关的控制器是否还在生命周期内，这意味着它仍然可以被访问，并没有被 WDDOEXIT 方法终止
IS_CHILD_NODE_RECURSIVE	查询 Context 子节点是否是递归或没有递归
IS_FINALIZED	确认节点信息和可能的子节点是否能被变更
IS_INITIALIZE_LEAD_SELECTION	判断运行时是否动态初始化头选择或始终设置为头选择。这相当于在开发环境中选择“Initialization Lead Selection”属性
IS_MANDATORY IS_MULTIPLE	确定节点元素的数量。在开发环境中，Context 节点属性使用这两种方法建立基数（参见 Context 节点属性中的相关章节） IS_MANDATORY 查询当前 Context 节点是否具有至少一个节点元素。如果是真（True），则该节点具有基数 1 ..1 或 1 到 n，即至少有一个元素被创建 IS_MULTIPLE 查询当前 Context 节点可以具有多个节点的元素。也就是，如果基数为 0 .. n 或 1 到 n，则返回真（True）
IS_MANDATORY_SELECTION IS_MULTIPLE_SELECTION	确定视图中节点元素被选择元素的数目。在开发环境中，这两种方法确定选择基数 IS_MANDATORY_SELECTION 指定一个节点元素必须始终选择，即该节点具有选择基数 1 .. 1 或 1 到 n IS_MULTIPLE_SELECTION 指定可以选择几个节点元素，即选择基数为 0 .. n 或 1 到 n
IS_MAPPING_COMPLETE	确定外部映射是否被执行，或者组件的映射是否被调用者执行
IS_SINGLETON	确定 Context 节点被定义为一个 Singleton 节点，一个 Singleton 节点只有在运行时其父节点被选中头选择时才存在。这允许 Web Dynpro 运行时，Context 可以优化内存结构

3）ADD 方法：允许添加子节点或节点的属性，见表 A-5。

表 A-5

方　　法	描　　述
ADD_ATTRIBUTE	添加的动态属性。这相当于用开发环境中的 Context 菜单创建属性。创建的结构字段 NAME（TYPE_NAME 或 RTTI）是非常重要的
ADD_CHILD_NODE	添加已有节点到节点树。这只能用于曾经从树中删除过节点，否则使用 ADD_NEW_CHILD_NODE 这相当于在开发环境中用 Context 菜单创建节点
ADD_NEW_CHILD_NODE	创建一个节点，并将其附加到节点树。在控制器中的节点名称必须是唯一的。如果指定 DDIC 结构的节点，则并不需要单独添加各个属性 – 框架直接生成属性 如果想删除该节点，则在创建节点时设置 IS_STATIC 为 False 是很重要的
ADD_NEW_MAPPED_CHILD_NODE	创建一个节点映射到另一个节点。这里只有名称和映射路径是相关的。程序员不需要创建任何属性。属性会从原来节点的属性自动复制 如果指定的映射路径，它是一个标准的映射。如果路径设置为初始值，需要进行外部可映射节点的定义。要完成映射，还需要调用方法 SET_MAPPING_COMPLETE
ADD_RECURSIVE_CHILD_NODE	添加一个递归的子节点。这相当于在开发环境中使用 Context 菜单创建递归节点

4）SET 方法：用来设置 Context 节点或子节点的属性，见表 A-6。

表 A-6

方　　法	说　　明
SET_ATTRIBUTE_FORMAT_PROPS	为属性设置特殊的格式的特性
SET_ATTRIBUTE_REFERENCE_FIELD	对于包含单位和数量或货币数量和货币单位的字段，可以在方法中写入。同一个节点只有一个字段被接受。 如果引用 ABAP 字典中所使用的结构定义的节点，它用在这里单独指定。
SET_ATTRIBUTE_REF_CALLBACK	应用程序可以使用此方法确定声明可引用货币和数量的字段为该元素的引用和属性。如果几个价格条目可能为一个货币，这种设置是重要的。
SET_ATTRIBUTE_VALUE_HELP	动态设置输入帮助，即 Context 属性分配的输入帮助可以动态改变。可以更改输入帮助类型（ABAP 字典搜索帮助、OVS 输入帮助、可编程输入帮助组件）以及指定的输入帮助名称
SET_ATTRIBUTE_VALUE_SET	设置允许值的列表
SET_FINALIZED	动态设置属性，不能在开发环境中设置。节点信息是不允许再次被改变的。它是恒定的。只要节点已设置此状态，它就不再改变，一旦改变这个节点，一个异常就被触发
SET_MAPPING_COMPLETE	完成外部可映射的节点的定义。该方法被用于外部 Context 映射。使用的组件定义了一个节点是外部映射和调用组件必须指定使用哪个节点。为此，它在组件使用的接口控制器中使用，并定义控制器和原始节点的路径
SET_NULLABLE	声明一个属性或所有属性用初始值和 NULL 值的方式来区分

5）REMOVE Methods 方法：可将 Context 节点或子节点的属性删除，见表 A-7。

注：不能删除已被定义为静态的属性。只能删除在运行时创建的属性。

表 A-7

方　　法	描　　述
REMOVE_ATTRIBUTE	动态属性可以反复使用这种方法，将其所有节点（运行时实例是无效的）的信息引用删除
REMOVE_CHILD_NODE	删除一个子节点，如果它没有被声明为非静态节点（见 ADD_NEW_CHILD_NODE），则静态创建的节点信息不会被删除
REMOVE_DYNAMIC_ATTRIBUTES	删除动态声明的所有属性。在此节点上所有节点的信息是无效的。

（4）IF_WD_CONTEXT_ELEMENT

接口 IF_WD_CONTEXT_ELEMENT 通过一个单一的 Context 元素进行赋值或取值。

大部分的该接口的方法也包含在 IF_WD_CONTEXT_NODE 中，主要用这些头选择的元素，或指定索引的元素。

方法：

1）GET_ATTRIBUTE：返回一个 Context 属性的值的方法。该方法的先决条件是应用程序知道该属性的数据类型。给定的属性的值被复制到导出参数。要做到这一点必须有导出参数用于值的写入。

如果属性名称<attr_name>：NOT，则它假设该值是布尔值而且相反的值被写入。

如果属性名称<attr_name>：FINAL，则系统会检查是否该属性值设置为最终并将这些信息写入。

如果属性不存在或值不能被写入，则异常类型 CX_WD_CONTEXT 被触发。

注：如果可以返回几个声明的元素的属性值，则使用方法 GET_STATIC_ATTRIBUTE 通常是更有效的。

2）GET_ATTRIBUTE_REF：该方法类似于 GET_ATTRIBUTE，但在以下方面是不同的：

- 返回参数代替输出参数，而且包含属性值的数据与引用。
- 引用的值不是存储在内部的引用。创建相同类型的数据，该值即被复制，然后该数据作为参数类型的引用输出。

3）GET_INDEX：返回当前元素的索引。

4）GET_CHILD_NODE：该方法返回具有指定名称的子节点。子节点总是由框架创建，而不是应用程序。

注：出于性能方面的考虑，子节点名称需指定大写字母。

5）GET_CHILD_NODES：该方法返回这一种元素的所有子节点作为一个表，其中包含名称和引用。

6）GET_NODE：该方法返回的元素节点的引用。如果元素是无效的，则一个例外（类型 CX_WD_CONTEXT）会被触发。因此该元素的有效性被确定。

7）GET_STATIC_ATTRIBUTES：该方法返回所有声明为静态属性的副本。如果结构是不相同的，则操作 move-corresponding 将被执行。这是个效率不高赋值，但在许多情况下，仍然是有用的。

静态属性是那些已被声明为静态或属于被声明为静态节点的结构。

8）GET_PATH：该方法返回控制器中原节点中元素（也就是说，不是在控制器中声明的映射）的路径。可以指定包含路径中的控制器的名称或不指定（见 IF_WD_CONTEXT_NODE=> GET_PATH）。

如果该路径是在搜索一个特定的控制器（例如，如在视图中绑定的路径）的方法，则方法 CL_WD_CONTEXT_SERVICES=> GET_ELEMENT_PATH_FOR_CTRL 必须使用，由此可以提高系统的性能。

9）GET_STATIC_ATTRIBUTES_REF：使用 GET_STATIC_ATTRIBUTES 时适用，如果是 GET_ATTRIBUTE，则用 GET_ ATTRIBUTE_REF。

10）IS_ALIVE：该方法检查相关的控制器是否还在生命周期内，这意味着它仍然可以被访问，并没有被 WDDOEXIT 方法终止。

11）IS_CHANGED_BY_CLIENT：指示 Context 元素是否被用户改变。

12）SET_CHANGED_BY_CLIENT：一个元素被用户改变（见 Context 更改日志）。

13）SET_ATTRIBUTE：按照属性名称设置属性值。

14）SET_STATIC_ATTRIBUTES：如果有多个静态声明的属性要赋值，建议使用 SET_STATIC_ATTRIBUTES 方法。

15）SET_ATTRIBUTE_NULL：设置属性为 NULL，值为 NULL 的属性总是显示为空。这使得可以区分空值和初始值。

16）SET_STATIC_ATTRIBUTES：设置所有静态声明的属性。如果不赋值，则静态属性将被设定为初始值。如果赋值即使两个结构不完全符合，MOVE-CORRESPONDING 也将执行。

17）SET_STATIC_ATTRIBUTES_NULL：将所有属性设置为特殊 NULL 值（参见方法：SET_ATTRIBUTE_NULL）。

18）SET_FINALIZED：一个元素设置为最终不允许用户或应用程序改变。最终设置不能

被撤销。但是可以删除这个元素。

19）SET_ATTRIBUTE_FINALIZED：只有一个属性被设置为最终类。

```
IS_FINALIZED
IS_ATTRIBUTE_FINALIZED
```

20）TO_XML：这个元素作为 XML 序列化的属性。

21）IS_SELECTED：确定此元素是否被选中（即这并不意味着必须头选择）。

```
IS_ATTRIBUTE_NULL
```

22）SET_SELECTED：选择元素。

3．个性化 APIs

下面的 Web 接口提供了个性化的 Web Dynpro ABAP 应用：

IF_WD_PERSONALIZATION 接口包含 Web Dynpro ABAP 组件在运行时返回相关配置信息或更改配置所使用的方法的集合。

方法：

1）SET_CONFIG_KEY：设置保存配置密钥键的方法。

2）GET_STATE：得到当前加载状态的信息。

3）LOAD_CONFIG_BY_KEY：用于加载配置或个性化的方法。

4）GET_VARIANTS：得到一个配置变式列表的方法。

5）ENTER_CHANGE_MODE：切换改变模式的方法。

6）SAVE：保存个性化数据的方法。

7）DELETE：删除个性化数据的方法。

8）GET_VALID_PERS_SCOPE：获得最大的保存范围的方法。

9）GET_DEFAULT_VARIANT：取得个性化默认值的方法。

10）RESTRICT_IMPL_PERS：用于限制隐式个性化的方法。

11）UPDATE_IMPL_PERS：在动态改变 UI 元素后更新隐式的个性化的方法。

12）GET_VARIANTS_FOR_SUBCOMPONENT：获取配置变式列表的方法。

13）SAVE_NEW_VARIANT：个性化数据的保存方法。

14）SET_DEFAULT_VARIANT：个性化设置的默认值的方法。

15）TRANSPORT_ALLOWED：用于指定是否允许数据传输的方法。

16）IS_IMPLEMENTATION_SUPPLIED：用于测试被指定的一个实例是否为一个组件使用的方法。

17）SUPPORT_DELTA_HANDLING：对于每个存储器动作，包括如最终用户在运行时删除按钮这样的操作，系统保存了隐含的个性化和明确的个性化的数据。但是，可以使用 SUPPORT_DELTA_HANDLING 指定系统只保存更改的明确的数据，而不是配置控制器的全部内容。

18）GET_TABLE_READER：提供接口读取配置数据表的方法。

4．门户集成 APIs

在门户网站中集成 Web Dynpro ABAP 的接口和类：

- IF_WD_PORTAL_INTEGRATION。

● CL_WDR_PORTAL_OBWEB_SERVICE。

（1）IF_WD_PORTAL_INTEGRATION

类 IF_WD_PORTAL_INTEGRATION 作为门户管理器提供门户网站与 Web Dynpro ABAP 应用程序集成的所有必要的方法。

方法：

1）NAVIGATE_TO_OBJECT：基于对象的导航（OBN）方法。

注：参数 USE_FORWARDPARAMETERS_OF_AL 确保可以从一个应用程序传输到下一个目标的 iView 中的 JavaScript。因此，不需要有任何 JavaScript 的目标 IView（return 'DynamicParameter='+objVal; ），所以参数从一个应用程序可以转发到下一个应用程序。如果应用设置开关 USE_FORWARDPARAMETERS_OF_AL＝abap_true，则应用程序自动集成到门户网站，一个额外的参数会自动插入执行该操作。然而，这并不与旧的 iView 中 JavaScript 相关联。

2）NAVIGATE_ABSOLUTE：绝对的导航方法。

3）NAVIGATE_RELATIVE：相对导航方法。

4）FIRE：触发门户网站的事件方法。

5）SUBSCRIBE_EVENT 和 UNSUBSCRIBE_EVENT：注册和取消注册事件的方法。

6）SET_APPLICATION_DIRTY_FLAG：设置“dirty”标志的方法。

7）SET_WORK_PROTECT_MODE：设置工作保护模式的方法。

（2）CL_WDR_PORTAL_OBNWEB_SERVICE

使用类 CL_WDR_PORTAL_OBNWEB_SERVICE，在 Web 服务支持下应用程序为基于对象导航接收到实际 Web 服务的封装，这个类可以更容易地使用接口，其包含数据已经被读取的高速缓存。

方法：

1）Constructor：不带参数，这个类可以被实例化。在这种情况下，标准的 Web 服务端口被使用。如果程序员为构造函数指定一个不同的端口，则该端口将被使用。

2）OBN_GET_DEFAULT_TARGET：该方法返回的 iView 的操作名称，显示名称和操作名称的默认操作。该方法需要有系统的别名和业务对象的名称。

注：使用的 iView 的名称，也可以触发在门户网站中的绝对导航。

3）OBN_GET_DEFAULT_TFO：该方法返回的 iView 的操作的名称，显示名称和操作名称的默认操作。该方法需要有系统别名的名称、业务对象和操作。

4）OBN_GET_TARGETS：该方法返回的所有可能的 iView 名称，显示名称的操作及有关的操作名称。该方法需要有系统的别名和业务对象的名称。使用这种方法，可以生成所有可能的操作的菜单，如为 LINK_TO_ACTION 生成菜单项。

5）OBN_GET_TFOS：该方法的工作原理同方法 OBN_GET_TARGETS。需要操作的名称。

6）OBN_IS_TARGET_EXIST：该方法告诉程序员一个依赖于系统别名的可能的目标和业务对象是否存在。

7）OBN_IS_TARGET_EXIST_TFO：该方法的工作原理与 OBN_IS_TARGET_EXIST 相同，但需要另外的操作名称。

附录 B　服务类（Service Classes）

以下服务类和接口提供 Web Dynpro 应用开发的一般服务：

- CL_WD_ADOBE_SERVICES。
- CL_WD_CONTEXT_SERVICES。
- CL_WD_DYNAMIC_TOOL。
- CL_WD_RUNTIME_SERVICES。
- CL_WD_UTILITIES。
- CL_WD_TRACE_TOOL 和 IF_WD_TRACE_TOOL。
- IF_WD_SELECT_OPTIONS。
- CL_WD_WEB_ICON。
- CL_WEB_DYNPRO。

1．CL_WD_ADOBE_SERVICES

类 CL_WD_ADOBE_SERVICES 可用于 ABAP 开发环境中，也可用于运行时的 Adobe 集成环境中。

该方法主要用于在运行时创建 PDF 文件，或解析运行时 XML 架构的表单接口、XML 格式的表单数据。

（1）方法

1）CREATE_PDF：该方法创建 PDF 文档，强制性参数有 ENABLED 和 DATA_SOURCE。

重要输入参数：

- ENABLED：指定的表单是否互动。
- DATA_SOURCE：Web Dynpro Context 的引用。

返回参数 PDF：

PDF 文件作为 XSTRING 输出。

2）GET_CABVERSION：为了保证表单互动功能，需要 ActiveX 控制框架（ACF）。该方法建立路径到 ACF，最终检查 ACF 是否是最新版本或更高版本是否存在。

（2）附加方法

其余的方法都是用来解析一个 XML 架构或 XML 数据。

通过 IXML 解析器，一个 XML 文档（该架构以表单形式存在）进行处理，一个内部表填充数据。作为架构表现为 Web Dynpro 的 Context，用于建立 Context 的所有信息保存在表中。如果该架构已创建一个 Web Dynpro 的 Context，则名称为 FROM_WD_CONTEXT 的元素（=True）可以在第一层级找到。在视图设计器中，程序员可以决定是否改变 Context，XML 架构和 XML 数据通常以同样的方式解析，唯一不同的是结果，这就是有两种不同的表类型的原因。

注：XML 数据和 XML 构架之间的差异：构架描述的是外部表单的数量和字段的类型，

而 XML 数据传输的实际数据包含在这些字段之中。出于这个原因架构是在设计时，数据指的是在运行时。

1）GET_CONTEXT：该方法返回描述 Web Dynpro Context 的内部表。

2）GET_DATA：该方法返回一个内部表，它表示 XML 数据。

3）GET_FROM_WD_CONTEXT：该方法搜索 XML 架构中所描述的 FROM_WD_CONTEXT 标志，并在 Web Dynpro Context 产生一个架构的情况下，返回字符 X，否则它返回一个空白字符。

（3）解析方法

1）PARSE_XML_SCHEMA：解析 XML 架构的方法。

2）PARSE_XML_DATA：解析 XML 数据的方法。

3）PARSE_SUBNODE 和 PARSE_XML_SUBNODE：解析深层嵌套层次结构的 XML 文档的方法。通过递归的方法调用。出于这个原因，每个 XML 文档都有自己的方法。

2．CL_WD_CONTEXT_SERVICES

CL_WD_CONTEXT_SERVICES 类用于识别 Context 的变化。这里指通过程序对 Context 操作的所有变化。通常情况下使用 Context 更改日志。

注：一个*属性名称意味着，所有节点的属性已经改变：一个 Context 节点的静态属性可能被改变，但它不能被指定为确定，如果设定这样的属性的值，则用户应调用实例方法 SET_STATIC_ATTRIBUTES。

（1）方法

1）SUBSCRIBE_TO_NODE_CHANGES：登记变更 Context 节点的方法。

2）UNSUBSCRIBE_FROM_NODE_CHANGES：注销变更 Context 节点的方法。

3）UNSUBSCRIBE_ALL：彻底注销变更的方法。

4）DEACTIVATE_SUBSCRIPTIONS：暂时停用注册变更的方法。

5）ACTIVATE_SUBSCRIPTIONS：重新注册变更的方法。

6）GET_CHANGE_LIST：取得带有可选复位变更列表的方法。

7）RESET_CHANGE_LIST：重新更改列表的方法。

（2）附加方法

下面的方法提供了映射和其原始节点的信息。它们主要用于视图中动态创建绑定路径及映射节点。

1）GET_NODE_METAPATH_FOR_CTRL。

2）GET_MAPPED_NODE_FOR_CONTROLLER。

3）GET_ORIGINAL_NODE。

4）GET_ELEMENT_PATH_FOR_CTRL。

注：以下的方法已经过时，不应该使用：UPDATE_NODE_FROM_TABLE_BY_KEY、UPDATE_ELEMENT_FROM_STRUCTURE 、 UPDATE_NODE_FROM_TABLE_BY_IND 、GET_TABLE_FROM_NODE。

（3）操作

处理 Context 变化：

1）生成的类 CL_WD_CONTEXT_SERVICES 的一个实例。

2）使用 SUBSCRIBE_TO_NODE_CHANGES 为一组 Context 的节点注册。

3）使用 GET_CHANGE_LIST 定期取得更改列表。

4）如果有必要，改变开关记录使用 ACTIVATE_SUBSCRIPTIONS 暂时关闭并重新打开。

5）删除变化记录的登记。

3．CL_WD_DYNAMIC_TOOL

类 CL_WD_DYNAMIC_TOOL 包含各种方法，这些方法对动态编程的第一个步骤（如 SE16 复制）是很有用的。

主要方法

1）IS_STRUCTURE_OR_TABLE：方法 SE16 复制。

2）GET_TABLE_DATA：读取数据的通用方法，所以没有任何 checkman 错误会发生。

3）CREATE_FORM_FROM_NODE：从一个节点的属性创建表单的方法。可以使用该方法作为复制模板。

4）CREATE_LAYOUT_DATA：根据父类型创建一个布局 UI 元素的实例的方法。可以使用该方法，在布局父类型不明的情况下，进行动态 UI 编程。

5）CREATE_NODEINFO_FROM_STRUCT：使用预定义的数据结构创建一个节点的信息对象的方法。可以使用该方法进行 SE16 复制。

注：使用 IF_WD_CONTEXT_NODE_INFO 的方法可以动态创建节点。

6）CREATE_TABLE_FROM_NODE：该方法的作用与 CREATE_FORM_FROM_NODE 相同，这里用于创建表格。

7）FILL_NODE：该方法已过时，不再使用。

8）SET_TREE_LEAD_SELECTION：该方法用于设置当前选定的叶或树中的节点。如果另一个元素已被选中，则该节点选中将被取消。如果没有 Context 元素设定，则当前选定的叶或节点被取消。当一棵树 UI 元素被使用时，在树的 Context 初始化后显示之前该方法被调用一次。如果它不是这种情况，则在同一时间选择几片叶子或节点取决于 Context 设置。

9）CREATE_RANGE_TABLE：该方法从指定的类型动态地创建一个范围表，如可以使用选择选项（Select Options）。然而，应用程序不仅限于选择选项，因为一个标准的范围表被创建时，使用 data mytable type range of mytype 语句创建 mytable。

4．CL_WD_RUNTIME_SERVICES

这个类提供了一般的 Web Dynpro 运行时服务。

方法：

ATTACH_FILE_TO_RESPONSE：不使用 UI 元素文件下载（File Download）进行文件导出方法。

5．CL_WD_UTILITIES：类 CL_WD_UTILITIES 包含了各种一般功能的方法。

方法：

1）CALC_WIDTH4DDLBBYKEY：使用项目列表计算 DropDownByKey UI 元素的宽度的方法。

2）SET_WIDTH_4DDLBBYKEY：使用项目列表计算和设定 DropDownByKey UI 元素的宽度的方法。

3）CONSTRUCT_WD_URL：计算 Web Dynpro 应用程序的 URL 的方法。

4）GET_OTR_TEXT_BY_ALIAS：提供 OTR 文本的 OTR 别名（系统调用包装器）的一个方法。

6．CL_WD_TRACE_TOOL 和 IF_WD_TRACE_TOOL

使用 CL_WD_TRACE_TOOL 类，该类指向接口 IF_WD_TRACE_TOOL，通过它的静态属性 INSTANCE，应用程序开发人员可以提供额外的数据到 Web Dynpro 跟踪工具。

要做到这一点，需使用接口 IF_WD_TRACE_TOOL 包含方法 ADD_TRACE_DATA。

示例：

```
IF CL_WD_TRACE_TOOL=>INSTANCE IS BOUND.
  DATA TRACE_DATA TYPE XSTRING.
  " <<< collect your application-specific trace data

  " Add your collected trace data to the trace file for

  " the current roundtrip
  CL_WD_TRACE_TOOL=>INSTANCE->ADD_TRACE_DATA(
    NAME = 'YourDomain/FileName.Ext'
    XSTRING = TRACE_DATA
  ).
ENDIF.
```

SAP 建议添加一个应用领域作为跟踪文件的前缀，如 SRM/。

7．IF_WD_SELECT_OPTIONS

IF_WD_SELECT_OPTIONS 接口用于集成在应用程序组件的选择屏幕上的 Selection Options。

方法：

（1）选择字段的方法

1）ADD_SELECTION_FIELD：选择屏幕中插入字段的主要方法。

2）ADD_SELECTION_FIELDS：选择屏幕中添加多个字段的方法。

3）GET_SELECTION_FIELD：返回选择屏幕上的字段的数据。

4）GET_SELECTION_FIELDS：返回选择屏幕上的所有字段的数据。

5）RESET_SELECTION_FIELD：复位范围（Range）表选择字段。

6）RESET_ALL_SELECTION_FIELDS：复位所有范围表选择字段。

7）CHECK_ALL_SELECTION_FIELDS：检查所有选择字段输入的正确性。

8）UPD_SELECTION_FIELD：更新一个选择字段的设置。

9）GET_RANGE_TABLE_OF_SEL_FIELD：读取一个范围表。

10）SET_RANGE_TABLE_OF_SEL_FIELD：设置一个范围表。

（2）布局的方法

插入布局元素的方法如下：

- ADD_HORIZONTAL_DIVIDER 和 GET_HORIZONTAL_DIVIDER：添加及返回选择屏幕中水平分割的数据。
- ADD_HORIZONTAL_DIVIDERS 和 GET_HORIZONTAL_DIVIDERS：添加及返回选择画面中多个水平分割的数据。
- ADD_TEXT_LINE 和 GET_TEXT_LINE：添加及返回选择屏幕中文本行中的数据（如字符串）。所有允许的设计是一个 TextView 元素。
- ADD_TEXT_LINES 和 GET_TEXT_LINES：新增及返回选择画面中的多行文本（字符串）。

（3）块（Blocks）的方法

与块相关的方法如下：ADD_BLOCK、ADD_BLOCKS、GET_BLOCK、GET_BLOCKS。

程序员可以使用的用户界面元素有组（Group）、托盘（Tray）或透明容器（Transparent Container）。

（4）参数的方法

这些方法对应于上述指定的选择字段的方法，区别在于参数的差异：ADD_PARAMETER_FIELD、ADD_PARAMETER_FIELDS、GET_PARAMETER_FIELD、GET_PARAMETER_ FIELDS、SET_VALUE_OF_PARAMETER_FIELD、GET_VALUE_OF_PARAMETER_FIELD、RESET_PARAMETER_FIELD、RESET_ALL_PARAMETER_FIELDS、CHECK_ALL_PARAMETER_FIELDS。

（5）标准结构的方法

标准结构的方法有 GET_SELECTION_SCREEN_ITEM、GET_SELECTION_SCREEN_ITEMS、ADD_SELECTION_SCREEN_ITEM、ADD_SELECTION_SCREEN_ITEMS、REMOVE_SELECTION_SCREEN_ITEM。

从选择画面中移除一个元素。

这里唯一的参数是要删除的字段的 ID（I_ID）。

- REMOVE_ALL_SEL_SCREEN_ITEMS

从选择屏幕中移除所有元素。如果有一个块，则连同其内容将整个块从显示屏幕中移除。

（6）帮助方法

1）CREATE_RANGE_TABLE：创建一个动态范围表。

2）SET_GLOBAL_OPTIONS：指定按钮是可见的。

8．CL_WD_WEB_ICON

这个类的静态方法提供有关 Web Icon 的信息。

方法：

1）GET_COLLECTIONS：该方法返回 Web Icon 集合。

2）GET_WEB_ICONS：该方法返回一个集合或所有集合的 Web Icon。

3）GET_WEB_ICON_URL：该方法返回 Web Dynpro 应用程序相关的 URL（一个 Web

Icon 的网址）。

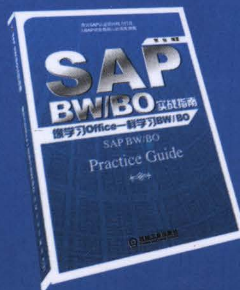
9．CL_WEB_DYNPRO

Web Dynpro 环境的应用程序是否使用 ABAP 字典搜索帮助在运行查询，可以通过类 CL_WEB_DYNRPO 的属性的 is_active 进行判断。

句法：

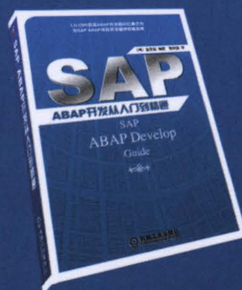
```
IF CL_WEB_DYNPRO=>is_active = abap_true.
  ...
ELSE.
...
ENDIF.
```

进阶阅读



《SAP BW/BO实战指南——像学习Office一样学习BW/BO》

ISBN.978-7-111-37555-5



《SAP ABAP开发从入门到精通》

ISBN 978-7-111-41700-2



《SAP 战略绩效管理完全解决方案》

ISBN 978-7-111-30582-8

ISBN 978-7-111-54749-5

主要读者对象：

- 企业内部的SAP ERP支持顾问；
- 企业内部的SAP ERP用户；
- SAP实施公司技术与业务顾问及开发顾问；
- 院校计算机科学与技术、软件工程、信息科学等专业的师生或相关培训机构。

本书主要介绍了与SAP Web开发技术——Web Dynpro For ABAP相关的集成应用和高级应用，还讲解了Web Dynpro For ABAP高端应用的各个技术难点中各元素之间的关联及制约关系。

本书结合实例讲解了在Web Dynpro组件中如何集成Flex，如何存储和访问文件，Web Dynpro如何在EP（Enterprise Portal）中展现及数据交互，如何继承Adobe Form这种应用极广的电子表格以及与之相关的数据的存取，广泛应用于SAP系统的ALV表格如何在Web Dynpro For ABAP中应用集成以及更多异于一般ALV列表的功能。此外，本书还介绍了在Web Dynpro程序中如何创建和使用辅助类，如何创建和使用服务调用，与用户交互时如何使用对话框、输入帮助、文本帮助及消息，在考虑比较复杂的Web Dynpro应用之间交互时如何使用Suspend和Resume调用等。

SAP

上架指导：ERP/SAP



51CTO学院
为中华崛起传播智慧

地址：北京市百万庄大街22号
邮政编码：100037

电话服务
服务咨询热线：010-88361066
读者购书热线：010-68326294
010-88379203
网络服务
机工官网：www.cmpbook.com
机工官博：weibo.com/cmp1952
金书网：www.golden-book.com
教育服务网：www.cmpedu.com
封面无防伪标均为盗版



机械工业出版社
微信公众号



计算机分社
微信订阅号

策划编辑◎丁诚 / 封面设计◎道乐文化

定价：59.00元